Composably secure data processing for Gaussian-modulated continuous variable quantum key distribution

Alexander G. Mountogiannakis[1], Panagiotis Papanastasiou[1], Boris Braverman[2], and Stefano Pirandola[1]
[1] *Department of Computer Science, University of York, York YO10 5GH, UK*
[2] *Xanadu, 372 Richmond St W, Toronto, M5V 2L7, Canada*

Continuous-variable (CV) quantum key distribution (QKD) employs the quadratures of a bosonic mode to establish a secret key between two remote parties, and this is usually achieved via a Gaussian modulation of coherent states. The resulting secret key rate depends not only on the loss and noise in the communication channel, but also on a series of data processing steps that are needed for transforming shared correlations into a final string of secret bits. Here we consider a Gaussian-modulated coherent-state protocol with homodyne detection in the general setting of composable finite-size security. After simulating the process of quantum communication, the output classical data is post-processed via procedures of parameter estimation, error correction, and privacy amplification. In particular, we analyze the high signal-to-noise regime which requires the use of high-rate (non-binary) low-density parity check codes. We implement all these steps in a Python-based library that allows one to investigate and optimize the protocol parameters to be used in practical experimental implementations of short-range CV-QKD.

I. INTRODUCTION

In quantum key distribution (QKD), two authenticated parties (Alice and Bob) aim at establishing a secret key over a potentially insecure quantum channel [1]. The security of QKD is derived from the laws of quantum mechanics, namely the uncertainty principle or the no-cloning theorem [2, 3]. In 1984, Bennett and Brassard introduced the first QKD protocol [4], in which information is encoded on a property of a discrete-variable (DV) system, such as the polarization of a photon. This class of protocols started the area of DV-QKD and their security has been extensively studied [1, 5].

Later on, at the beginning of the 2000s, a more modern family of protocols emerged, in which information is encoded in the position and momentum quadratures of a bosonic mode [6, 7]. These continuous-variable (CV) QKD protocols are particularly practical and cost-effective, being already compatible with the current telecommunication technology [1, 8]. CV-QKD protocols are very suitable to reach high rates of communication (e.g. comparable with the ultimate channel limits [9]) over distances that are compatible with metropolitan areas [10]. Recently, CV-QKD protocols have also been shown to reach very long distances, comparable to those of DV protocols [11, 12].

In a basic CV-QKD protocol, Alice encodes a classical variable in the coherent states of a bosonic mode via Gaussian modulation. These states are then transmitted to Bob via the noisy communication channel. At the output, Bob measures the received states via homodyne [6, 11] (or heterodyne [7]) detection in order to retrieve Alice's encoded information. In the worst-case scenario, all the loss and noise present in the channel is ascribed to a malicious party (Eve) who tries to intercept the states and obtain information about the key. By carefully estimating Eve's perturbation, Alice and Bob can apply procedures of error correction (EC) to remove noise from their data, and then privacy amplification (PA), which makes their error-corrected data completely secret.

In this work, we consider the basic CV-QKD protocol based on Gaussian modulation of coherent states and homodyne detection. Starting from a simulation of the quantum communication in typical noisy conditions, we process the generated data into a finite-size secret key which is composably secure against collective Gaussian attacks. Besides developing the technical procedure step-by-step, we implement it in an open-access Python library associated with this paper [13]. In particular, our procedure for data processing is based on the composable secret key rate developed in Ref. [14, Sec. III].

In order to address a short-range regime with relatively high values of the signal to noise ratio (SNR), the step of EC is based on high-rate (non-binary) low-density parity check codes (LDPC) [15–19], whose decoding is performed by means of a suitable iterative sum-product algorithm [16, 20] (even though the min-sum algorithm [21] can be used as an alternative). After EC, the procedure of PA is based on universal hash functions [22]. Because of the length of the generated strings can be substantial, techniques with low complexity are preferred, so that we use the Toeplitz-based hash function, which is simple, parallelizable and can moreover be accelerated by using the Fast Fourier Transform (FFT) [23].

The paper is structured as follows. In Sec. II we start with the description of the coherent-state protocol including a realistic model for the detector setup. After an initial analysis of its asymptotic security, we discuss the steps of parameter estimation (PE), EC and PA, ending with the composable secret key rate of the protocol. In Sec. III, we go into further details of the protocol simulation and data processing, also presenting the pseudocode for the entire process. In Sec. IV we provide numerical results that are obtained by our Python library in simulated experiments. Finally, Sec. V is for conclusions.

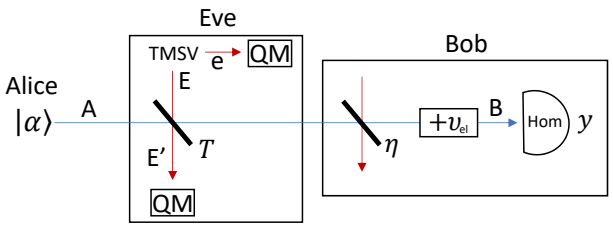

FIG. 1: Structure of a CV-QKD protocol with (Gaussian-modulated) coherent states considering a receiver that might have trusted levels of inefficiency and electronic noise. In the middle, the thermal-loss channel is induced by the collective Gaussian attack of Eve, who uses a beam splitter with transmissivity T and a TMSV state with variance ω. Eve's outputs are stored in quantum memories (QM) for a later quantum measurement, whose optimal performance is bounded by the Holevo information.

II. COHERENT-STATE PROTOCOL AND SECURITY ANALYSIS

A. General description and mutual information

Assume that Alice prepares a bosonic mode A in a coherent state $|\alpha\rangle$ whose amplitude is Gaussian-modulated (see Fig. 1 for a schematic). In other words, we may write $\alpha = (q + ip)/2$, where $x = q, p$ is the mean value of the generic quadrature $\hat{x} = \hat{q}, \hat{p}$, which is randomly chosen according to a zero-mean Gaussian distribution $\mathcal{N}(0, \sigma_x^2)$ with variance $\sigma_x^2 := \mu - 1 \geq 0$, i.e., according to the Gaussian probability density

$$p_{\mathcal{N}}(x) = (\sqrt{2\pi}\sigma_x)^{-1} \exp\left[-x^2/(2\sigma_x^2)\right]. \tag{1}$$

Note that we have $q = 2\,\mathrm{Re}\,\alpha$ and $p = 2\,\mathrm{Im}\,\alpha$ (in fact, we use the notation of Ref. [8, Sec. II], where $[\hat{q}, \hat{p}] = 2i$ and the vacuum state has noise-variance equal to 1). Then, σ_x^2 represents the modulation variance while μ is the total signal variance (including the vacuum noise).

The coherent state is sent through an optical fiber of length L, which can be modelled as a thermal-loss channel with transmissivity $T = 10^{-\frac{AL}{10}}$ (e.g., $A = 0.2$ dB/km) and thermal noise $\omega = 2\bar{n} + 1$, where $\bar{n}$ is the thermal number associated with an environmental mode E. The process can equivalently be represented by a beam splitter with transmissivity T mixing Alice's mode A with mode E, which is in a thermal state with $\bar{n}$ mean photons. The environmental thermal state can be purified in a two-mode squeezed vacuum state (TMSV) Φ_{eE}, i.e., a Gaussian state for modes e and E, with zero mean and covariance matrix (CM) [8]

$$\mathbf{V}_{eE}(\omega) = \begin{pmatrix} \omega\mathbf{I} & \sqrt{\omega^2-1}\mathbf{Z} \\ \sqrt{\omega^2-1}\mathbf{Z} & \omega\mathbf{I} \end{pmatrix}, \tag{2}$$

where $\mathbf{I} := \mathrm{diag}(1, 1)$ and $\mathbf{Z} := \mathrm{diag}(1, -1)$. This dilation based on a beamsplitter and a TMSV state is known as an "entangling cloner" attack and represents a realistic collective Gaussian attack [24] (see Fig. 1). Recall that the class of collective Gaussian attacks is optimal for Gaussian-modulated CV-QKD protocols [1].

At the other end of the channel, Bob measures the incoming state using a homodyne detector with quantum efficiency η and electronic noise υ_{el} (we may also include local coupling losses in parameter η). The detector is randomly switched between the two quadratures. Let us denote by $\hat{y}$ the generic quadrature of Bob's mode B just before the (ideal) homodyne detector. Then, the outcome y of the detector satisfies the input-output formula

$$y = \sqrt{T\eta}x + z, \tag{3}$$

where z is (or approximately is) a Gaussian noise variable with zero mean and variance

$$\sigma_z^2 = 1 + \upsilon_{\mathrm{el}} + \eta T\xi, \tag{4}$$

with υ_{el} being the electronic noise of the setup and ξ is channel's excess noise, defined by

$$\xi := \frac{1-T}{T}(\omega - 1). \tag{5}$$

Here it is important to make two observations. The first consideration is about the general treatment of the detector at Bob's side, where we assume the potential presence of trusted levels of quantum efficiency and electronic noise. In the worst-case scenario, these levels can be put equal to zero and assume that these contributions are implicitly part of the channel transmissivity and excess noise. In other words, one has $T\eta \to T$ in Eq. (3), while υ_{el} becomes part of ξ in Eq. (4), so that $\xi \to \xi + \upsilon_{\mathrm{el}}/T$ in Eq. (5). The second point is that there might be other imperfections in Alice's and Bob's setups that are not mitigated or controllable by the parties (e.g., modulation and phase noise). These imperfections are automatically included in the channel loss and noise via the general relations of Eqs. (3) and (4). Furthemore, the extra noise contributions can be considered to be Gaussian in the worst-case scenario, resorting to the optimality of Gaussian attacks in CV-QKD [1].

Assuming the general scenario in Fig. 1, let us compute Alice and Bob's mutual information. From Eq. (3), we calculate the variance of y, which is equal to

$$\begin{aligned} \sigma_y^2 &= T\eta\sigma_x^2 + \sigma_z^2 \\ &= T\eta(\mu - 1 + \xi) + 1 + \upsilon_{\mathrm{el}}. \end{aligned} \tag{6}$$

For the conditional variance, we compute

$$\sigma_{y|x}^2 = \sigma_y^2(\mu = 1) = \eta T\xi + 1 + \upsilon_{\mathrm{el}}. \tag{7}$$

The mutual information associated with the CVs x and y is given by the difference between the differential entropy

$h(y)$ of y and the conditional entropy $h(y|x)$, i.e.,

$$I(x:y) = h(y) - h(y|x) = \frac{1}{2}\log_2\left(\frac{\sigma_y^2}{\sigma_{y|x}^2}\right) \tag{8}$$

$$= \frac{1}{2}\log_2\left[1 + \frac{\mu - 1}{\xi + (1+\upsilon_{\rm el})/T\eta}\right] \tag{9}$$

$$= \frac{1}{2}\log_2\left[1 + \mathrm{SNR}\right]. \tag{10}$$

The mutual information has the same value no matter if it is computed in direct reconciliation (DR), where Bob infers Alice's variable, or reverse reconciliation (RR), where Alice infers Bob's variable.

As we can see from the expression above, the mutual information contains the signal-to-noise (SNR) term

$$\mathrm{SNR} = (\mu - 1)/\mathcal{X}, \tag{11}$$

where

$$\mathcal{X} = \xi + (1+\upsilon_{\rm el})/T\eta \tag{12}$$

is known as equivalent noise. The joint Gaussian distribution of the two variables x and y has zero mean and the following CM

$$\mathbf{\Sigma}_{xy} = \begin{pmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{pmatrix} \tag{13}$$

where $\rho = \mathbb{E}(xy)/\sigma_x\sigma_y$ is a correlation parameter (here $\mathbb{E}$ denotes the expected value). From the classical formula $I(x:y) = -(1/2)\log_2(1-\rho^2)$ [25], we see that

$$\rho = \sqrt{\frac{\mathrm{SNR}}{1+\mathrm{SNR}}}. \tag{14}$$

It is important to stress that, in order to asymptotically achieve the maximum number of shared bits $I(x:y)$ per channel use in RR, Bob needs to send $h(y|x)$ bits through the public channel according to Slepian-Wolf coding [26, 27]. In practice, Alice and Bob will perform a suboptimal procedure of EC/reconciliation revealing more information $\mathrm{leak}_{\rm EC} \geq h(y|x)$. To account for this, one assumes that only a portion $\beta I(x:y)$ of the mutual information can be achieved by using the reconciliation parameter $\beta \in [0,1)$. This is crucial parameter that depends on various technical quantities as we will see later.

In the ideal case of perfect reconciliation ($\beta = 1$), the mutual information between the parties depends monotonically on the SNR. However, in a realistic case where the reconciliation is not efficient ($\beta < 1$), the extra leaked information $(1-\beta)I(x:y)$ also depends on the SNR. Therefore, by trying to balance the terms $\beta I(x:y)$ and $\mathrm{leak}_{\rm EC} = h(y|x) + (1-\beta)I(x:y)$, one finds an optimal value for the SNR and, therefore, an optimal value for the total signal variance μ. In practice, this optimal working point can be precisely computed only after T and ξ are estimated through the procedure of PE (which is discussed later in Sec. II C). However, a rough estimate of this optimal value can be obtained by an educated guess of the channel parameters (e.g., the approximate transmissivity could be guessed from the length of the fiber and the expected standard loss rate).

B. Asymptotic key rate

Let us compute the secret key rate that the parties would be able to achieve if they could use the quantum communication channel an infinite number of times. Besides $\beta I(x:y)$, we need to calculate Eve's Holevo information $\chi(E:y)$ on Bob's outcome y. This is in fact the maximum information that Eve can steal under the assumption of collective attacks, where she perturbs the channel in a independent and identical way, while storing all her outputs in a quantum memory (to be optimally measured at the end of the protocol). It is important to stress that, for any processing done by Bob $y \to y'$, we have $\chi(E:y') \leq \chi(E:y)$ so that the latter value can always be taken as an upper bound for the actual eavesdropping performance.

Let us introduce the entanglement-based representation of the protocol, where Alice's input ensemble of coherent states is generated on mode A by heterodyning mode A' of a TMSV $\Phi_{A'A}$ with variance μ. Note that this representation is not strictly necessary in our analysis (which may be carried over in prepare and measure completely), but we adopt it anyway for completeness, so as to give the total state with all the correlations between Alice, Bob and Eve.

The output modes A' and B shared by the parties will be in a zero-mean Gaussian state $\rho_{A'B}$ with CM [8]

$$\mathbf{V}_{A'B} = \begin{pmatrix} \mu\mathbf{I} & c\mathbf{Z} \\ c\mathbf{Z} & b\mathbf{I} \end{pmatrix}, \tag{15}$$

where we have set

$$c := \sqrt{T\eta(\mu^2-1)}, \tag{16}$$

$$b := T\eta(\mu+\xi) + 1 - T\eta + \upsilon_{\rm el}. \tag{17}$$

Then, the global output state $\rho_{A'BeE'}$ of Alice, Bob and Eve is zero-mean Gaussian with CM

$$\mathbf{V}_{A'BeE'} = \begin{pmatrix} \mu\mathbf{I} & c\mathbf{Z} & \mathbf{0} & \zeta\mathbf{Z} \\ c\mathbf{Z} & b\mathbf{I} & \gamma\mathbf{Z} & \theta\mathbf{I} \\ \mathbf{0} & \gamma\mathbf{Z} & \omega\mathbf{I} & \psi\mathbf{Z} \\ \zeta\mathbf{Z} & \theta\mathbf{I} & \psi\mathbf{Z} & \phi\mathbf{I} \end{pmatrix}, \tag{18}$$

where $\mathbf{0}$ is the 2×2 zero matrix and we set

$$\gamma := \sqrt{\eta(1-T)(\omega^2-1)}, \tag{19}$$

$$\zeta := -\sqrt{(1-T)(\mu^2-1)}, \tag{20}$$

$$\theta := \sqrt{\eta T(1-T)}(\omega-\mu), \tag{21}$$

$$\psi := \sqrt{T(\omega^2-1)}, \tag{22}$$

$$\phi := T\omega + (1-T)\mu. \tag{23}$$

To compute the Holevo bound, we need to derive the von Neumann entropies $S(\rho_{eE'})$ and $S(\rho_{eE'|y})$ which can be computed from the symplectic spectra of the reduced CM $\mathbf{V}_{eE'}$ and the conditional CM $\mathbf{V}_{eE'|y}$. Setting

$$\mathbf{C} = \begin{pmatrix} \gamma\mathbf{Z} & \theta\mathbf{I} \end{pmatrix}, \ \mathbf{\Pi} := \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \tag{24}$$

we may use the pseudo-inverse to compute [8]

$$\begin{aligned} \mathbf{V}_{eE'|y} &= \mathbf{V}_{eE'} - \mathbf{C}^T[\mathbf{\Pi}(b\mathbf{I})\mathbf{\Pi}]^{-1}\mathbf{C} && (25)\\ &= \mathbf{V}_{eE'} - b^{-1}\mathbf{C}^T\mathbf{\Pi}\mathbf{C} && (26)\\ &= \begin{pmatrix} \omega\mathbf{I} & \psi\mathbf{Z} \\ \psi\mathbf{Z} & \phi\mathbf{I} \end{pmatrix} - b^{-1}\begin{pmatrix} \gamma^2\mathbf{\Pi} & \gamma\theta\mathbf{\Pi} \\ \gamma\theta\mathbf{\Pi} & \theta^2\mathbf{\Pi} \end{pmatrix}. && (27) \end{aligned}$$

Since both $\mathbf{V}_{eE'}$ and $\mathbf{V}_{eE'|y}$ are two-mode CMs, it is easy to compute their symplectic spectra, that we denote by $\{\nu_\pm\}$ and $\{\tilde{\nu}_\pm\}$, respectively. Their general analytical expressions are too cumbersome to show here unless we take the limit $\mu \gg 1$. For instance, we have $\nu_+ \to (1-T)\mu$ and $\nu_- \to \omega$.

Finally, we may write the Holevo bound as

$$\begin{aligned} \chi(E:y) &= S(\rho_{eE'}) - S(\rho_{eE'|y}) \\ &= h(\nu_+) + h(\nu_-) - h(\tilde{\nu}_+) - h(\tilde{\nu}_-), \end{aligned} \tag{28}$$

where

$$h(\nu) := \frac{\nu+1}{2}\log_2\frac{\nu+1}{2} - \frac{\nu-1}{2}\log_2\frac{\nu-1}{2}. \tag{29}$$

The asymptotic key rate is given by

$$\begin{aligned} R_{\text{asy}} &= \beta I(x:y) - \chi(E:y) && (30)\\ &= R(\beta, \mu, \eta, \upsilon_{\text{el}}, T, \xi). && (31) \end{aligned}$$

Suppose that η and υ_{el} are known and the parties have preliminary estimates of T and ξ. Then, for a target β, Alice can compute an optimal value μ_{opt} for the signal variance μ by optimizing the asymptotic rate in Eq. (31).

C. Parameter estimation

In a realistic implementation of the protocol, the parties use the quantum channel a finite number N of times. The first consequence of this finite-size scenario is that their knowledge of the channel parameters T and ξ is not perfect. Thus, once the quantum communication is over, Alice and Bob declare m random instances $\{x_i\}$ and $\{y_i\}$ of their local variables x and y. From these instances, they build the maximum-likelihood estimators $\widehat{T}$ of the transmissivity T and $\widehat{\Xi}$ of the excess noise variance $\Xi := \eta T\xi$, for which they also exploit their knowledge of the trusted levels of detector/setup efficiency η and electronic noise υ_{el}.

Following Ref. [28], we write

$$\widehat{T} = \frac{1}{\eta\sigma_x^4}\left(\widehat{C}_{xy}\right)^2 \tag{32}$$

where

$$\widehat{C}_{xy} = \frac{1}{m}\sum_{i=1}^{m} x_i y_i, \tag{33}$$

is the estimator for the covariance $C_{xy} = \sqrt{\eta T}\sigma_x^2$ between x and y. This covariance is normally distributed with following mean and variance

$$\begin{aligned} \mathbb{E}(\widehat{C}_{xy}) &= \sqrt{\eta T}\sigma_x^2 = C_{xy}, && (34)\\ \text{Var}(\widehat{C}_{xy}) &= \frac{1}{m}\left(2\eta T\left(\sigma_x^2\right)^2 + \sigma_x^2\sigma_z^2\right) := V_{C_{xy}}. && (35) \end{aligned}$$

Then one can express $\widehat{T}$ as a (scaled) non-central chi-squared variable

$$\widehat{T} = \frac{V_{C_{xy}}}{\eta^2\left(\sigma_x^2\right)^2}\left(\frac{\widehat{C}_{xy}}{\sqrt{V_{C_{xy}}}}\right)^2, \tag{36}$$

since $\widehat{C}_{xy}/\sqrt{V_{C_{xy}}}$ follows a standard normal distribution. The mean and variance of $\widehat{T}$ is given by the associated noncentral chi-squared parameters $k=1$ and $\lambda = \frac{C_{xy}^2}{V_{C_{xy}}}$. Therefore, we obtain

$$\mathbb{E}(\widehat{T}) = \frac{V_{C_{xy}}}{\eta^2\left(\sigma_x^2\right)^2}\left(1 + \frac{C_{xy}^2}{V_{C_{xy}}}\right), \tag{37}$$

and

$$\text{Var}(\widehat{T}) = \frac{2V_{C_{xy}}^2}{\eta^4\left(\sigma_x^2\right)^4}\left(1 + 2\frac{C_{xy}^2}{V_{C_{xy}}}\right). \tag{38}$$

Using Eqs. (34) and (35), and keeping only the significant terms with respect to $1/m$, we have

$$\mathbb{E}(\widehat{T}) = T, \ \text{Var}(\widehat{T}) = \frac{4}{m}T^2\left(2 + \frac{\sigma_z^2}{\eta T\sigma_x^2}\right) := \sigma_T^2. \tag{39}$$

The estimator for the noise variance σ_z^2 is given by

$$\widehat{\sigma_z^2} = \frac{1}{m}\sum_{i=1}^{m}\left(y_i - \sqrt{\eta\widehat{T}}x_i\right)^2. \tag{40}$$

Assuming $\hat{T} \approx T$ and rescaling by $1/\sigma_z$ the term inside the brackets in the relation above, we get a standard normal distribution for the variable $\frac{z_i}{\sigma_z}$ with $z_i = y_i - \sqrt{\eta T}x_i$. Thus, we obtain

$$\widehat{\sigma_z^2} = \frac{\sigma_z^2}{m}\sum_{i=1}^{m}\left(\frac{z_i}{\sigma_z}\right)^2, \tag{41}$$

and observe that this is a (scaled) chi-squared variable. From the associated chi-squared parameters $k=m$ (and $\lambda=0$), we calculate the following mean and variance

$$\begin{aligned} \mathbb{E}(\widehat{\sigma_z^2}) &= \sigma_z^2, && (42)\\ \text{Var}(\widehat{\sigma_z^2}) &= \frac{2\left(\sigma_z^2\right)^2}{m}. && (43) \end{aligned}$$

Then, from the formula

$$\sigma_z^2 = 1 + v_{\rm el} + \Xi, \tag{44}$$

we have that

$$\widehat{\Xi} = \widehat{\sigma_z^2} - v_{\rm el} - 1, \tag{45}$$

$$\mathbb{E}(\widehat{\Xi}) = \Xi, \tag{46}$$

$$\mathrm{Var}(\widehat{\Xi}) = \frac{2\left(\sigma_z^2\right)^2}{m} := \sigma_{\widehat{\Xi}}^2. \tag{47}$$

For m sufficiently large, and up to an error probability $\varepsilon_{\rm PE}$, the channel parameters fall in the intervals

$$T \in [\widehat{T} - w\sigma_{\widehat{T}}, \widehat{T} + w\sigma_{\widehat{T}}], \tag{48}$$

$$\Xi \in [\widehat{\Xi} - w\sigma_{\widehat{\Xi}}, \widehat{\Xi} + w\sigma_{\widehat{\Xi}}], \tag{49}$$

where $\sigma_{\widehat{T}}$, $\sigma_{\widehat{\Xi}}$ are given by the Eq. (39) and (47), where we replace the actual values T and σ_z^2 with their corresponding estimators. Note that w is expressed in terms of $\varepsilon_{\rm PE}$ via the inverse error function, i.e.,

$$w = \sqrt{2}\,\mathrm{erf}^{-1}(1 - \varepsilon_{\rm PE}). \tag{50}$$

The worst-case estimators will be given by

$$T_m = \widehat{T} - w\sigma_{\widehat{T}}, \;\; \Xi_m = \widehat{\Xi} + w\sigma_{\widehat{\Xi}} \tag{51}$$

so we have $T \geq T_m$ and $\Xi \leq \Xi_m$ up to an error $\varepsilon_{\rm PE}$ (see Appendix A for other details on the derivation of T_m).

In the next step, Alice and Bob compute an overestimation of Eve's Holevo bound in terms of T_m and Ξ_m, so that they may write the modified rate

$$R_m := \beta I(x:y) - \chi(E:y)|_{T_m,\Xi_m}. \tag{52}$$

Accounting for the number of signals sacrificed for PE, the actual rate in terms of bits per channel use is given by the rescaling

$$R_m \to \frac{n}{N} R_m, \tag{53}$$

where $n = N - m$ are the instances for key generation.

Note that, from the estimators $\widehat{T}$ and $\widehat{\Xi}$, the parties may compute an estimator for the SNR, i.e.,

$$\widehat{\rm SNR} = \frac{(\mu-1)\eta\widehat{T}}{1 + v_{\rm el} + \widehat{\Xi}}. \tag{54}$$

Therefore, in a more practical implementation, the rate in Eq. (52) is replaced by the following expression

$$R_m = \beta\, I(x:y)|_{\widehat{T},\widehat{\Xi}} - \chi(E:y)|_{T_m,\Xi_m}, \tag{55}$$

$$I(x:y)|_{\widehat{T},\widehat{\Xi}} = \frac{1}{2}\log_2\left(1 + \widehat{\rm SNR}\right). \tag{56}$$

For a fixed β, the parties could potentially optimize the SNR over the signal variance μ. In fact, before the quantum communication, they may use rough estimates about transmissivity and excess noise to compute value $\mu_{\rm opt}(\beta)$ that maximizes the asymptotic key rate.

In a practical implementation, the data generated by the QKD protocol is sliced in $n_{\rm bks} \gg 1$ blocks, each block being associated with the quantum communication of N points (modes). Assuming that the channel is sufficiently stable over time, the statistics (estimators and worst-case values) can be computed over

$$M := m n_{\rm bks} \gg m \tag{57}$$

random instances, so that all the estimators (i.e, $\widehat{T}$, $\widehat{\Xi}$, $\widehat{\rm SNR}$, T_m, and Ξ_m) are computed over M points and we replace $R_m \to R_M$ in Eq. (55), i.e., we consider

$$R_M = \beta\, I(x:y)|_{\widehat{T},\widehat{\Xi}} - \chi(E:y)|_{T_M,\Xi_M}, \tag{58}$$

This also means that we consider an average of $m = M/n_{\rm bks}$ points for PE in each block, and an average number of $n = N - M/n_{\rm bks}$ key generation points from each block to be processed in the step of EC. Because N is typically large, the variations around the averages can be considered to be negligible, which means that we may assume m and n to be the actual values for each block.

Finally note that, if the channel instead varies over a timescale comparable to the block size N (which is a condition that may occur in free-space quantum communications [14, 29]), then we may need to perform PE independently for each block. In this case, one would have a different rate for each block, so that the final key rate will be given by an average. However, for ground-based fiber-implementations, the channel is typically stable over long times, which is the condition assumed here.

D. Error correction

Once PE has been done, the parties process their remaining n pairs (key generation points) in a procedure of EC. Here we combine elements from various works [16, 19, 21, 30, 31]. The procedure can be broken down in steps of normalization, discretization, splitting, LDPC encoding/decoding, and EC verification.

1. *Normalization*

In each block of size N, Alice and Bob have n pairs $\{x_i, y_i\}$ of their variables x and y that are related by Eq. (3) and can be used for key generation. As a first step, Alice and Bob normalize their variables by dividing them by the respective standard deviations, i.e., [21]

$$x \to X := x/\sigma_x, \quad y \to Y := y/\sigma_y, \tag{59}$$

so X and Y have the following CM

$$\mathbf{\Sigma}_{XY} = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}. \tag{60}$$

Variables X and Y follow a standard normal bivariate distribution with correlation $\rho = \mathbb{E}(XY)$, which is connected to the SNR by Eq. (14), where the latter is approximated by Eq. (54) in a practical scenario.

Under conditions of stability for the quantum communication channel, the normalization in Eq. (59) is performed over the entire record of $nn_{\rm bks}$ key generation points. Using the computed standard deviations σ_x and σ_y, we then build the strings $X^n = x^n/\sigma_x$ and $Y^n = y^n/\sigma_y$ for each block, starting from the corresponding n key generation points $x^n = \{x_i\}$ and $y^n = \{y_i\}$.

2. Discretization

Bob discretizes his normalized variable Y in a p-ary variable K with generic value $\kappa \in \{0, \ldots, 2^p - 1\}$ being an element of a Galois field $\mathcal{GF}(2^p)$ (see Appendix B). This is achieved as follows. As a first step, he sets a cut-off α such that $|Y| \leq \alpha$ occurs with negligible probability, which is approximately true for $\alpha \geq 3$. Then, Bob chooses the size $\delta = 2\alpha 2^{-p}$ of the intervals (bins) $[a_\kappa, b_\kappa)$ of his lattice, whose border points are given by [19]

$$a_\kappa = \begin{cases} -\infty & \text{for} \quad \kappa = 0, \\ -\alpha + \kappa\delta & \text{for} \quad \kappa > 0, \end{cases} \tag{61}$$

and

$$b_\kappa = \begin{cases} -\alpha + (\kappa+1)\delta & \text{for } \kappa < 2^p - 1, \\ \infty & \text{for } \kappa = 2^p - 1. \end{cases} \tag{62}$$

Finally, for any value of $Y \in [a_\kappa, b_\kappa)$, Bob takes K equal to κ. Thus, for n points, the normalized string Y^n is transformed into a string of discrete values K^n. Note that this discretization technique is very basic. Indeed one could increase the performance by adopting bins of different sizes depending on the estimated SNR.

3. Splitting

Bob sets an integer value for $q < p$ and computes $d = p - q$. Then, he splits his discretized variable in two parts $K = (\overline{K}, \underline{K})$, where the top variable $\overline{K}$ is q-ary and the bottom variable $\underline{K}$ is d-ary. Their values are defined by splitting the generic value κ in the following two parts

$$\overline{\kappa} = \frac{\kappa - (\kappa \mod 2^d)}{2^d}, \quad \underline{\kappa} = (\kappa \mod 2^d). \tag{63}$$

In other words, we have

$$\kappa = \overline{\kappa} 2^d + \underline{\kappa}. \tag{64}$$

With the top variable $\overline{K}$, Bob creates 2^q super bins with each super bin containing 2^d bins associated with the bottom variable $\underline{K}$. See also Fig. 2.

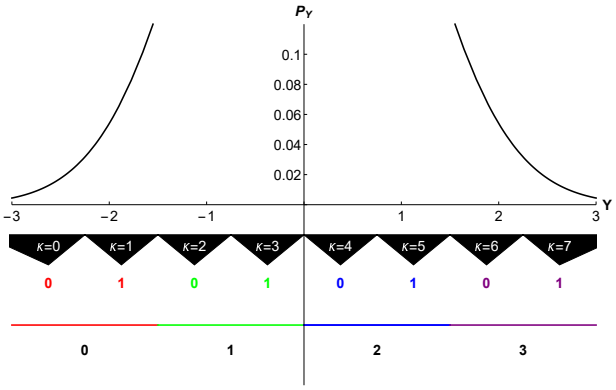

FIG. 2: Discretization and splitting with $\alpha = 3$, $p = 3$ and $q = 2$. The variable Y follows a normal distribution P_Y so that the probability of $|Y| > 3$ is assumed to be negligible. Variable Y and the bins defined in Eqs. (61) and (62) identify a discrete variable K with values $\kappa = 0, \ldots, 7$ (black triangles). During the splitting stage, each bin can be described by two numbers: $\overline{\kappa} = 0 \ldots, 3$ associated with $q = 2$, and $\underline{\kappa} = 0, 1$ associated with $d = p - q = 1$. We see that 2^d bins belong to each super bin $\overline{\kappa}$ (colored intervals). Bob uses the parity check matrix of a non-binary LDPC code to encode the information $\overline{\kappa}$ related to the super bin while the position $\underline{\kappa}$ of the bin inside the super bin is broadcast through the public channel.

Repeating this for n points provides a string of values $\overline{K}^n$ for the super bins and another string for the relative bin-positions $\underline{K}^n$. The most significant string $\overline{K}^n$ is locally processed by an LDPC code (more details below), while the least significant string $\underline{K}^n$ is side information that is revealed through the public channel.

4. LDPC encoding and decoding

Bob constructs an $l \times n$ parity check matrix $\mathbf{H}$ with q-ary entries from $\mathcal{GF}(2^q)$ according to Ref. [30]. This matrix is applied to the top string $\overline{K}^n$ to derive the l-long syndrome $K^l_{\rm sd} = \mathbf{H}\overline{K}^n$, where the matrix-vector product is defined in $\mathcal{GF}(2^q)$. The syndrome is sent to Alice together with the direct communication of the bottom string $\underline{K}^n$. The parity check matrix is associated with an LDPC code [16], which may encode $k = n - l$ source symbols into n output symbols, so that it has rate

$$R_{\rm code} := k/n = 1 - R_{\rm syn}, \;\; R_{\rm syn} := l/n. \tag{65}$$

What value of $R_{\rm code}$ to use and, therefore, what matrix $\mathbf{H}$ to build is explained afterwards in this subsection.

From the knowledge of the syndrome $K^l_{\rm sd}$, Bob's bottom string $\underline{K}^n$ and her local string X^n, Alice decodes Bob's top string $\overline{K}^n$. This is done via an iterative belief propagation algorithm [16] where, in every iteration, she updates a codeword likelihood function (see also Appendix C). The initial likelihood function before any it-

eration comes from the a priori probabilities

$$P(\overline{K}|X,\underline{K}) = \frac{P(\overline{K},\underline{K}|X)}{\sum_{\overline{K}} P(\overline{K},\underline{K}|X)}, \tag{66}$$

where

$$P(\overline{K},\underline{K}|X) = P(K|X) \tag{67}$$

$$= \frac{1}{2}\mathrm{erf}\left(\frac{b_\kappa - \rho x/\sigma_x}{\sqrt{2(1-\rho^2)}}\right) - \frac{1}{2}\mathrm{erf}\left(\frac{a_\kappa - \rho x/\sigma_x}{\sqrt{2(1-\rho^2)}}\right). \tag{68}$$

At any iteration $\leq \mathrm{iter}_{\max}$ Alice finds the argument that maximizes the likelihood function. If the syndrome of this argument is equal to K^l_{sd}, then the argument forms her guess $\widehat{K}^n$ of Bob's top string $\overline{K}^n$. However, if this syndrome matching test (SMT) is not satisfied within the maximum number of iterations $\mathrm{iter}_{\max}$, then the block is discarded. The possibility of failure in the SMT reduces the total number of input blocks from n_{bks} to $n_{\mathrm{SMT}} = p_{\mathrm{SMT}} n_{\mathrm{bks}}$ where p_{SMT} is the probability of successful matching within the established $\mathrm{iter}_{\max}$ iterations.

Let us compute the LDPC rate R_{code} to be used. First notice how Alice and Bob's mutual information decreases as a consequence of the classical data processing inequality [25] applied to the procedure. We have

$$I(x:y) \geq I(X:Y) \tag{69}$$

$$\geq I(X:K) = H(K) - H(K|X) \tag{70}$$

$$\geq H(K) - \mathrm{leak}_{\mathrm{EC}}, \tag{71}$$

where $\mathrm{leak}_{\mathrm{EC}} \geq H(K|X)$ comes from the Wolf-Slepian limit [26, 27] and $H(K)$ is the Shannon entropy of K, which can be computed over the entire record of $n_{\mathrm{ent}} = n n_{\mathrm{bks}}$ key generation points (under stability conditions). The parties empirically estimate the entropy $H(K)$. To do so, they build the MLE

$$\widehat{H}(K) = -\sum_{\kappa=0}^{2^p-1} f_\kappa \log_2 f_\kappa, \tag{72}$$

where $f_\kappa = n_\kappa/n_{\mathrm{ent}}$ stands for the frequencies of the symbols κ, defined as the ratio between the number of times n_κ that a symbol appears in the string over the string's length n_{ent}. For this estimator, it is easy to show that [32]

$$H(K) \geq \widehat{H}(K) - \delta_{\mathrm{ent}}, \tag{73}$$

with penalty

$$\delta_{\mathrm{ent}} = \log_2(n_{\mathrm{ent}})\sqrt{\frac{2\log(2/\varepsilon_{\mathrm{ent}})}{n_{\mathrm{ent}}}}. \tag{74}$$

This bound is valid up to an error probability $\varepsilon_{\mathrm{ent}}$.

In Eq. (71), the leakage $\mathrm{leak}_{\mathrm{EC}}$ is upper-bounded by the equivalent number of bits per use that are broadcast after the LDPC encoding in each block, i.e.

$$\mathrm{leak}_{\mathrm{EC}} \leq d + R_{\mathrm{syn}} q. \tag{75}$$

Therefore, combining the two previous bounds, we may write

$$I(x:y) \geq \beta I(x:y) := \widehat{H}(K) - \delta_{\mathrm{ent}} + R_{\mathrm{code}} q - p. \tag{76}$$

Note that in a practical implementation Alice and Bob do not access $I(x:y)$, but rather $I(x:y)|_{\widehat{T},\widehat{\Xi}}$ from Eq. (56). Therefore, considering this modification in Eq. (76), we more precisely write

$$\beta I(x:y)|_{\widehat{T},\widehat{\Xi}} = \widehat{H}(K) - \delta_{\mathrm{ent}} + R_{\mathrm{code}} q - p, \tag{77}$$

so the rate in Eq. (58) becomes

$$R^{\mathrm{EC}}_M = \widehat{H}(K) - \delta_{\mathrm{ent}} + R_{\mathrm{code}} q - p - \chi(E:y)|_{T_M,\Xi_M}. \tag{78}$$

From Eqs. (56) and (77), we see that the LDPC code must be chosen to have rate

$$R_{\mathrm{code}} \simeq q^{-1}\left[\frac{\beta}{2}\log_2\left(1+\widehat{\mathrm{SNR}}\right) + p - \widehat{H}(K) + \delta_{\mathrm{ent}}\right] \tag{79}$$

for some estimated SNR and key entropy. A value of the reconciliation efficiency β is acceptable only if we can choose parameters $\alpha \geq 3$ and q [33], such that $R_{\mathrm{code}} \leq 1$. Once R_{code} is known, the sparse parity check matrix $\mathbf{H}$ of the LDPC code can be constructed following Ref. [30].

5. *Verification*

An important final step in the EC procedure is the verification of the n_{SMT} error-corrected blocks that have successfully passed the SMT. For each of these blocks, the parties possess two n-long q-ary strings with identical syndromes, i.e., Bob's top string $\overline{K}^n$ and Alice's guess $\widehat{K}^n$. The parties convert their strings into a binary representation, $\overline{K}^n_{\mathrm{bin}}$ and $\widehat{K}^n_{\mathrm{bin}}$, so that each of them is qn bit long. In the next step, Alice and Bob compute t-bit long hashes of their converted binary strings following Ref. [31] (universal hash functions are used because of their resistance to collisions). In particular, they set $t = \lceil -\log_2 \varepsilon_{\mathrm{cor}} \rceil$, where $\varepsilon_{\mathrm{cor}}$ is known as 'correctness'.

Then, Bob discloses his hash to Alice, who compares it with hers. If the hashes are identical, the verification stage is deemed successful. The two strings $\overline{K}^n_{\mathrm{bin}}$ and $\widehat{K}^n_{\mathrm{bin}}$ are identical up to a small error probability $2^{-t} \leq \varepsilon_{\mathrm{cor}}$. In such a case, the associated bottom string $\underline{K}^n$ held by both parties is converted by both parties into binary $\underline{K}^n_{\mathrm{bin}}$ and appended to the respective strings, i.e., $\overline{K}^n_{\mathrm{bin}} \to \overline{K}^n_{\mathrm{bin}}\underline{K}^n_{\mathrm{bin}}$ and $\widehat{K}^n_{\mathrm{bin}} \to \widehat{K}^n_{\mathrm{bin}}\underline{K}^n_{\mathrm{bin}}$. Such binary concatenations are promoted to the next step of PA.

By contrast, if the hashes are different, then the two postprocessed strings are discarded (together with the public bottom string). Therefore, associated with the hash verification test, we have a probability of success

that we denote by $p_{\rm ver}$. This is implicitly connected with $\varepsilon_{\rm cor}$. In fact, if we decrease $\varepsilon_{\rm cor}$, we increase the length of the hashes to verify, meaning that we increase the probability of spotting an uncorrected error in the strings, leading to a reduction of the success probability $p_{\rm ver}$.

Thus, combining the possible failures in the syndrome and hash tests, we have a total probability of success for EC, which is given by

$$p_{\rm EC} = p_{\rm SMT} p_{\rm ver} := 1 - p_{\rm FER}, \tag{80}$$

where $p_{\rm FER}$ is known as 'frame error rate' (FER).

Note that the effective value of $p_{\rm EC}$ depends not only on $\varepsilon_{\rm cor}$ but also on the specific choice for the LDPC code. In particular, in the presence of a very noisy channel, using a high-rate LDPC code (equivalent to a high value of β) implies low correction performances and, therefore, a low value for $p_{\rm EC}$ (SMT and/or hash test tend to fail). By contrast, the use of a low-rate EC code (low value of β) implies a high value for $p_{\rm EC}$ (good performance for both SMT and hash test). Thus, there is an implicit trade-off between $p_{\rm EC}$ and β. For fixed values of $\varepsilon_{\rm cor}$ and β, the value of $p_{\rm EC}$ (success of the EC test) still depends on the channel noise, so that it needs to be carefully calculated from the experimental/simulation points.

E. Privacy amplification and composably-secure finite-size key

The final step is that of PA, after which the secret key is generated. Starting from the original $n_{\rm bks}$ blocks (each being N-size), the parties derive $p_{\rm EC} n_{\rm bks}$ successfully error-corrected binary strings

$$S := \overline{K}^n_{\rm bin} \underline{K}^n_{\rm bin} \simeq \widehat{S} := \widehat{K}^n_{\rm bin} \underline{K}^n_{\rm bin}, \tag{81}$$

each string containing np bits. In this final step, all the surviving error-corrected strings are compressed into shorter strings that are decoupled from Eve (up to a small error probability that we discuss below). This compression step can be implemented on each sequence individually, or globally on a concatenation of sequences. Here, the latter approach is adopted and is certainly valid under conditions of channel stability.

Concatenating their local $p_{\rm EC} n_{\rm bks}$ error-corrected strings, Alice and Bob construct two long binary sequences $\mathbf{S} \simeq \widehat{\mathbf{S}}$, each having $\tilde{n} := p_{\rm EC} n_{\rm bks} np$ bits. Each of these sequences will be compressed to a final secret key of $r := p_{\rm EC} n_{\rm bks} n \tilde{R}$ bits, where $\tilde{R}$ is determined by the composable key rate (see below). The compression is achieved via universal hashing by applying a Toeplitz matrix $\mathbf{T}_{r,\tilde{n}}$ which is calculated efficiently with the use of FFT (more details in Appendix D). Thus, from their sequences, Alice and Bob finally retrieve the secret key

$$\mathbf{K} = \mathbf{T}_{r,\tilde{n}} \mathbf{S} \simeq \mathbf{T}_{r,\tilde{n}} \widehat{\mathbf{S}}. \tag{82}$$

An important parameter to consider for the step of PA is the 'secrecy' $\varepsilon_{\rm sec}$, which bounds the distance between the final key and an ideal key from which Eve is completely decoupled. Technically, one further decomposes $\varepsilon_{\rm sec} = \varepsilon_{\rm s} + \varepsilon_{\rm h}$, where $\varepsilon_{\rm s}$ is a smoothing parameter $\varepsilon_{\rm s}$ and $\varepsilon_{\rm h}$ is a hashing parameter. These PA epsilon parameters are combined with the parameter from EC.

Let us call $\tilde{\rho}^n$ the classical-classical-quantum state shared by Alice, Bob and Eve after EC. We may write

$$p_{\rm EC} D(\tilde{\rho}^n, \rho_{\rm id}) \leq \varepsilon := \varepsilon_{\rm sec} + \varepsilon_{\rm cor}, \tag{83}$$

where ε is the epsilon security of the protocol, D is the trace distance and $\rho_{\rm id}$ is the output of an ideal protocol where Eve is completely decoupled from Bob, with Alice's and Bob's keys being exactly the same [14, App. G].

Accounting for the estimation of the channel parameters [cf. Eq. (51)] and the key entropy [cf. Eq. (73)] is equivalent to replacing $\tilde{\rho}^n$ with a worst-case state $\tilde{\rho}^n_{\rm wc}$ in the computation of the key rate. However, there is a small probability $\varepsilon'_{\rm PE}$ that we have a different state $\rho_{\rm bad}$ violating one or more of the tail bounds associated with the worst-case estimators. This means that, on average, we have the state

$$\rho_{\rm PE} = (1 - \varepsilon'_{\rm PE}) \tilde{\rho}^n_{\rm wc} + \varepsilon'_{\rm PE} \rho_{\rm bad}. \tag{84}$$

Because we impose $p_{\rm EC} D(\tilde{\rho}^n_{\rm wc}, \rho_{\rm id}) \leq \varepsilon$ and the previous equation implies $D(\tilde{\rho}^n_{\rm wc}, \rho_{\rm PE}) \leq \varepsilon'_{\rm PE}$, the triangle inequality provides $p_{\rm EC} D(\rho_{\rm PE}, \rho_{\rm id}) \leq \varepsilon + p_{\rm EC} \varepsilon'_{\rm PE}$, meaning that the imperfect parameter estimation adds an extra term $p_{\rm EC} \varepsilon'_{\rm PE}$ to the epsilon security of the protocol. In other words, we have that the protocol is secure up to re-defining $\varepsilon \to \varepsilon + p_{\rm EC} \varepsilon'_{\rm PE}$. Note that we have

$$\varepsilon'_{\rm PE} = (1 - 2\varepsilon_{\rm PE}) \varepsilon_{\rm ent} + (1 - \varepsilon_{\rm ent}) 2\varepsilon_{\rm PE} + 2\varepsilon_{\rm PE} \varepsilon_{\rm ent} \tag{85}$$

$$\simeq 2\varepsilon_{\rm PE} + \varepsilon_{\rm ent}, \tag{86}$$

so we can write

$$\varepsilon = \varepsilon_{\rm sec} + \varepsilon_{\rm cor} + p_{\rm EC} \varepsilon'_{\rm PE} \tag{87}$$

$$\simeq \varepsilon_{\rm s} + \varepsilon_{\rm h} + \varepsilon_{\rm cor} + p_{\rm EC}(2\varepsilon_{\rm PE} + \varepsilon_{\rm ent}). \tag{88}$$

A typical choice is to set $\varepsilon_{\rm s} = \varepsilon_{\rm h} = \varepsilon_{\rm cor} = \varepsilon_{\rm PE} = \varepsilon_{\rm ent} = 2^{-32} \simeq 2.3 \times 10^{-10}$, so that $\varepsilon \lesssim 4 \times 10^{-9}$ for any $p_{\rm EC}$.

For success probability $p_{\rm EC}$ and ε-security against collective (Gaussian) attacks, the secret key rate of the protocol (bits per channel use) takes the form [14, Eq. (105)]

$$R = \frac{n p_{\rm EC}}{N} \tilde{R}, \;\; \tilde{R} := \left(R^{\rm EC}_M - \frac{\Delta_{\rm AEP}}{\sqrt{n}} + \frac{\Theta}{n} \right), \tag{89}$$

where $R^{\rm EC}_M$ is given in Eq. (78) and the extra terms are equal to the following [34]

$$\Delta_{\rm AEP} := 4 \log_2 \left(2^{p/2} + 2 \right) \sqrt{\log_2 \left(\frac{18}{p^2_{\rm EC} \varepsilon^4_{\rm s}} \right)}, \tag{90}$$

$$\Theta := \log_2 [p_{\rm EC}(1 - \varepsilon^2_{\rm s}/3)] + 2 \log_2 \sqrt{2} \varepsilon_{\rm h}. \tag{91}$$

Note that the discretization bits p appear in $\Delta_{\rm AEP}$ [35]

The practical secret key rate in Eq. (89) can be compared with a corresponding theoretical rate

$$R_{\text{theo}} = \frac{n\tilde{p}_{\text{EC}}}{N}\left(R_M^* - \frac{\tilde{\Delta}_{\text{AEP}}}{\sqrt{n}} + \frac{\tilde{\Theta}}{n}\right), \tag{92}$$

where $\tilde{p}_{\text{EC}}$ is guessed, with $\tilde{\Delta}_{\text{AEP}}$ and $\tilde{\Theta}$ being computed on that guess. Then, we have

$$R_M^* = \tilde{\beta}\, I(x:y)|_{T,\Xi} - \chi(E:y)|_{T_M^*,\Xi_M^*}, \tag{93}$$

where $\tilde{\beta}$ is also guessed, and the various estimators are approximated by their mean values, so that $\widehat{T} \simeq T$, $\widehat{\Xi} \simeq \Xi = \eta T \xi$ and we have set

$$T_M^* = T - w\sigma_T, \tag{94}$$
$$\Xi_M^* = \Xi + w\sigma_\Xi, \tag{95}$$

with w depending on ε_{PE} as in Eq. (50).

III. PROTOCOL SIMULATION AND DATA PROCESSING

Here we sequentially go over the steps of the protocol and its postprocessing, as they need to be implemented in a numerical simulation or an actual experimental demonstration. We provide more technical details and finally present the pseudocode of the entire procedure.

A. Main parameters

We start by discussing the main parameters related to the physical setup, communication channel and protocol. Some of these parameters are taken as input, while others need to be estimated by the parties, so that they represent output values of the simulation.

Setup: Main parameters are Alice's total signal variance μ, Bob's trusted levels of local efficiency η, and electronic noise υ_{el}. These are all input values.

Channel: Main parameters are the effective transmissivity T, and excess noise ξ. These are input values to our simulation, which are used to create the input-output relation of Eq. (3). In an experimental implementation, these values are generally unknown and Eq. (3) comes from the experimental data.

Protocol: Main parameters are the number of blocks n_{bks}, the size of each block N, the total number M of instances for PE, the various epsilon parameters $\varepsilon_{\text{s,h,...}}$ and the p-bit discretization, so the alphabet size is $D = 2^p$. These are all input values. Output values are the estimators $\widehat{T}$, $\widehat{\Xi}$, T_M, Ξ_M, the EC probability of success p_{EC}, the reconciliation parameter β, the final rate R and key sting $\mathbf{K}$.

In Tables I and II, we summarize the main input and output parameters. In Table III, we schematically show the formulas for other related parameters.

parameter	description
L	Channel length (km)
A	Attenuation rate (dB/km)
ξ	Excess noise
η	Detector/Setup efficiency
υ_{el}	Electronic noise
β	(Target) reconciliation efficiency
n_{bks}	Number of blocks
N	Block size
M	Number of PE runs
p	Discretization bits
q	Most significant (top) bits
α	Phase-space cut-off
$\text{iter}_{\max}$	Max number of EC iterations
$\varepsilon_{\text{PE, s, h, corr}}$	Epsilon parameters
μ	Total signal variance

TABLE I: Main input parameters

parameter	description
μ_{opt}	Optimal signal variance
R_{asy}	Asymptotic key rate
$\widehat{T}$, $\widehat{\Xi}$, T_M, Ξ_M	Channel estimators
$\widehat{\text{SNR}}$	Estimated SNR
$\widehat{H}(K)$	Key entropy estimator
R_{code}	Code rate
p_{EC}	EC success probability
fnd_{rnd}	EC syndrome matching round
r	Final key length
R	Composable key rate
$\mathbf{K}$	Final key
ε	ε-security

TABLE II: Main output parameters

B. Quantum communication

The process of quantum communication can be simplified in the following two steps before and after the action of the channel:

Preparation: Alice encodes Nn_{bks} instances $\{x_i\}$ of the mean x of the generic quadrature $\hat{x}$, such that $x \sim \mathcal{N}(0, \mu - 1)$. In the experimental practice, the two conjugate quadratures are independently encoded in the amplitude of a coherent state, but only one of them will survive after the procedure of sifting (here implicitly assumed).

Measurement: After the channel and the projection of (a randomly-switched) homodyne detection, Bob decodes Nn_{bks} instances $\{y_i\}$ of $y = \sqrt{T\eta}x + z$, where the noise variable $z \sim \mathcal{N}(0, \sigma_z^2)$ has variance σ_z^2 as in Eq. (4).

parameter	description	formula
T	Channel transmissivity	$10^{-AL/10}$
σ_z^2	Noise variance	$1+v_{\mathrm{el}}+\eta T\xi$
Ξ	Excess noise variance	$\eta T\xi$
ω	Thermal noise	$\frac{T\xi-T+1}{1-T}$
$\mathcal{X}$	Equivalent noise	$\xi+\frac{1+v_{\mathrm{el}}}{T\eta}$
SNR	Signal-to-noise ratio	$(\mu-1)/\mathcal{X}$
m	PE instances per block	M/n_{bks}
n	Key generation points per block	$N-m$
FER	Frame error rate	$1-p_{\mathrm{EC}}$
GF	Number of the $\mathcal{GF}$ elements	2^q
δ	Lattice step in phase space	$\frac{\alpha}{2^{p-1}}$
d	Least significant (bottom) bits	$p-q$
t	Verification hash output length	$\lceil -\log_2 \varepsilon_{\mathrm{cor}} \rceil$
ρ	Correlation coefficient	$\sqrt{\frac{\widehat{\mathrm{SNR}}}{1+\widehat{\mathrm{SNR}}}}$
δ_{ent}	Entropy penalty	See Eq. (74)
$\tilde{n}$	Total bit string length after EC	$n\ p\ n_{bks}\ p_{\mathrm{EC}}$

TABLE III: Related parameters

As mentioned above, it is implicitly assumed that Alice and Bob perform a sifting stage where Bob classically communicates to Alice which quadrature he has measured (so that the other quadrature is discarded).

C. Parameter estimation

The stage of PE is described by the following steps:

Random positions: Alice randomly picks M positions $i \in [1, Nn_{\mathrm{bks}}]$, say $\{i_u\}_{u=1}^M$. On average $m = M/n_{\mathrm{bks}}$ positions are therefore picked from each block, and $n = N-m$ points are left for key generation in each block (for large enough blocks, the spread around these averages is negligible).

Public declaration: Using a classical channel, Alice communicates the M pairs $\{i_u, x_{i_u}\}_{u=1}^M$ to Bob.

Estimators: Bob sets a PE error $\varepsilon_{\mathrm{PE}}$. From the pairs $\{x_{i_u}, y_{i_u}\}_{u=1}^M$, he computes the estimators $\widehat{T}$ and $\widehat{\Xi}$, and the worst-case estimators T_M and Ξ_M for the channel parameters (see formulas in Sec. II C).

Early termination: Bob checks the threshold condition $I(x:y)|_{\widehat{T},\widehat{\Xi}} > \chi(E:y)|_{T_M,\Xi_M}$, which is computed in terms of the estimators $\widehat{T}$ and $\widehat{\Xi}$, and worst-case estimators T_M and Ξ_M (associated with $\varepsilon_{\mathrm{PE}}$). If the threshold condition is not satisfied, then the protocol is aborted.

D. Error correction

The procedure of EC is performed on each block of size N and consists of the following steps:

Normalization: For key generation, Alice and Bob have n pairs $\{x_i, y_i\}$ of their variables x and y that are related by Eq. (3). As a first step, Alice and Bob normalize their variables x and y according to Eq. (59), therefore creating X and Y.

Discretization: Bob sets the cut-off value α and the step $\delta = \alpha 2^{1-p}$ of his lattice, whose generic bin $Y[a_\kappa, b_\kappa)$ is defined by Eqs. (61) and (62). Then, he discretizes his normalized variable Y into a p-ary variable K with generic value $\kappa \in \{0, \ldots, 2^p-1\}$ with the following rule: For any value of his variable $Y \in [a_\kappa, b_\kappa)$, Bob takes K equal to κ.

Splitting: Bob sets an integer value for q and computes $d = p-q$. From his discretized variable K, he creates the top q-ary variable $\overline{K}$ and the bottom d-ary variable $\underline{K}$, whose generic values $\overline{\kappa}$ and $\underline{\kappa}$ are defined by Eq. (63). For n points, he therefore creates a string $\overline{K}^n$ which is locally processed via an LDPC code (see below), and another string $\underline{K}^n$ which is revealed through the public channel.

LDPC encoding: From the SNR estimator $\widehat{\mathrm{SNR}}$, the entropy estimator $\widehat{H}(K)$, and a target reconciliation efficiency β, the parties use Eq. (79) to derive the rate R_{code} of the LDPC code. They then build its $l \times n$ parity-check matrix $\mathbf{H}$ with q-ary entries from $\mathcal{GF}(2^q)$, using the procedure of Ref. [30], i.e., (i) the column weight d_v (number of nonzero elements in a column) is constant and we set $d_v = 2$; (ii) the row weight d_c adapts to the formula $R_{\mathrm{code}} = 1 - d_v/d_c$ and is as uniform as possible; and (iii) the overlap (inner product between two columns) is never larger than 1. Once $\mathbf{H}$ is constructed, Bob computes the syndrome $K_{\mathrm{sd}}^l = \mathbf{H}\overline{K}^n$, which is sent to Alice together with the bottom string $\underline{K}^n$.

LDPC decoding: From the knowledge of the syndrome K_{sd}^l, Bob's bottom string $\underline{K}^n$ and her local string X^n, Alice decodes her guess $\widehat{\overline{K}}^n$ of Bob's top string $\overline{K}^n$. This is done via a sum-product algorithm [16], where in each iteration iter $<$ iter$_{\max}$ Alice updates a suitable likelihood function with initial value given by the a priori probabilities in Eq. (68). If the syndrome of $\widehat{\overline{K}}^n$ is equal to K_{sd}^l, then Alice's guess $\widehat{\overline{K}}^n$ of Bob's top string $\overline{K}^n$ is promoted to the next verification step. If the syndrome matching test fails for iter$_{\max}$ iterations, the block is discarded and the frequency/probability $1-p_{\mathrm{SMT}}$ of this event is registered. For more details of the sum-product algorithm see Appendix C.

Verification: Each pair of promoted strings $\overline{K}^n$ and $\widehat{K}^n$ is converted to binary $\overline{K}^n_{\rm bin}$ and $\widehat{K}^n_{\rm bin}$. Over these, the parties compute hashes of $t = \lceil -\log_2 \varepsilon_{\rm cor} \rceil$ bits. Bob discloses his hash to Alice, who compares it with hers. If the hashes are identical, the parties convert the corresponding bottom string $\underline{K}^n$ into binary $\underline{K}^n_{\rm bin}$ and promote the two concatenations

$$S := \overline{K}^n_{\rm bin}\underline{K}^n_{\rm bin} \simeq \widehat{S} := \widehat{K}^n_{\rm bin}\underline{K}^n_{\rm bin} \tag{96}$$

to the next step of PA. By contrast, if the hashes are different, then $\overline{K}^n_{\rm bin}$ and $\widehat{K}^n_{\rm bin}$ are discarded, together with $\underline{K}^n$. The parties compute the frequency/probability of success of the hash verification test $p_{\rm ver}$ and derive the overall success probability of EC $p_{\rm EC} = p_{\rm SMT}p_{\rm ver} = 1 - p_{\rm FER}$. In more detail, the strings $\overline{K}^n_{\rm bin}$ and $\widehat{K}^n_{\rm bin}$ are broken into Q-bit strings that are converted to Q-ary numbers forming the strings $\overline{K}^{n'}_Q$ and $\widehat{K}^{n'}_Q$ with $n' = nq/Q$ symbols for $Q > q$. In case nq/Q is not an integer, the strings $\overline{K}^n_{\rm bin}$ and $\widehat{K}^n_{\rm bin}$ are padded with sq zeros so that $n' = (n+s)q/Q \in \mathbb{N}$. Bob then derives independent uniform random integers $v_i = 1, \ldots, 2^{Q^*} - 1$, where v_i is odd, and an integer $u = 0, \ldots, 2^{Q^*} - 1$, for $i = 1, \ldots, n'$ and $Q^* \leq Q + t - 1$ with $\varepsilon_{\rm cor} \leq 2^{-t}$ being the target collision probability. After Bob communicates his choice of universal families to Alice, they both hash each of the Q-ary numbers and combine the results according to the following formula [31]

$$\tilde{h}\left(\mathbf{x}\right) = \left(\sum_{i=1}^{n'} v_i x_i\right) + u, \tag{97}$$

where $\mathbf{x} = \overline{K}^{n'}_Q$ (for Bob) or $\widehat{K}^{n'}_Q$ (for Alice). Summation and multiplication in Eq. (97) are modulo 2^{Q^*}. In practice, this is done by discarding the overflow (number of bits over Q^*) of $\tilde{h}\left(\mathbf{x}\right)$. Then they keep only the first t bits to form the hashes (where typically $t = 32$).

E. Privacy amplification

After EC, Alice and Bob are left with $p_{\rm EC}n_{\rm bks}$ successfully error-corrected binary strings, each of them being represented by Eq. (96) and containing np bits. By concatenation, they build two long binary sequences $\mathbf{S} \simeq \widehat{\mathbf{S}}$, each having $\tilde{n} := p_{\rm EC}n_{\rm bks}np$ bits. For the chosen level of secrecy $\varepsilon_{\rm sec}$, the parties compute the overall epsilon security ε from Eq. (88) and the key rate $R = \frac{np_{\rm EC}}{N}\tilde{R}$ according to Eq. (89). Finally, the sequences $\mathbf{S} \simeq \widehat{\mathbf{S}}$ are compressed into a final secret key of length $r := p_{\rm EC}n_{\rm bks}n\tilde{R}$ bits by applying a Toeplitz matrix $\mathbf{T}_{r,\tilde{n}}$, so the secret key is given by Eq. (82).

F. Pseudocode of the procedure

In Algorithm 1, we present the entire pseudocode of the procedure, whose steps are implemented in Python [13].

Algorithm 1 High-Level Routine Overview

$L, A, \eta, \xi, v_{\rm el}, n_{\rm bks}, N, M, \varepsilon_{\rm PE}, \varepsilon_s, \varepsilon_h, \varepsilon_{cor}, \beta, {\rm iter}_{\max}, p, q, \alpha \leftarrow$ Input_Values_Definition()
$T, \sigma_z^2, \Xi, m, n, t, {\rm GF}, \delta, d \leftarrow$ Dependent_Values()
Validity_Checks()
if is_mu_optimal **then**
    $\mu_{\rm opt} \leftarrow$ Optimal_Signal_Variance()
else
    $\mu \leftarrow$ user_input
end if
for blk $= 1, 2, \ldots n_{\rm bks}$ **do**
    x[blk] $\leftarrow$ State_Preparation()
    y[blk] $\leftarrow$ State_Transmission()
    y[blk] $\leftarrow$ State_Measurement()
    x[blk] $\leftarrow$ Key_Sifting()
end for
$R_{\rm asy}$, $I(x:y)|_{T,\Xi}$, $\chi(E:y)|_{T,\Xi} \leftarrow$ Rate_Calculation()
$\{i_u, x_{i_u}\}_{u=1}^M, \{i_u, y_{i_u}\}_{u=1}^M \leftarrow$Sacrificed_States_Selection()
$\hat{T}, \hat{\Xi}, T_M, \Xi_M \leftarrow$ Parameter_Estimation()
R_M, $I(x:y)|_{\hat{T},\hat{\Xi}}$, $\chi(E:y)|_{T_M,\Xi_M} \leftarrow$ Rate_Calculation()
if $I(x:y)|_{\hat{T},\hat{\Xi}} \leq \chi(E:y)|_{T_M,\Xi_M}$ **then**
    Abort_Protocol()
end if
$X, Y \leftarrow$ Normalization()
$\widehat{\rm SNR}, \rho \leftarrow$ Code_Estimations()
for blk $= 1, 2, \ldots n_{\rm bks}$ **do**
    K[blk] $\leftarrow$ Discretization()
    $\overline{K}$[blk], $\underline{K}$[blk] $\leftarrow$ Splitting()
    $p_{\overline{K}|X,\underline{K}}$[blk] $\leftarrow$ A_Priori_Probabilities_Calculation()
end for
$\widehat{H}$(K),$R_{\rm code} \leftarrow$ Code_Rate_Calculation()
$\mathbf{H} \leftarrow$ LDPC_Code_Generation()
for blk $= 1, 2, \ldots n_{\rm bks}$ **do**
    $\overline{K}^l_{\rm sd}$[blk] $\leftarrow$ Bob_Syndrome_Calculation()
    $\widehat{K}^n$[blk], fnd, ${\rm rnd}_{\rm fnd} \leftarrow$ Non_Binary_Decoding()
    $\widehat{K}^n_{\rm bin}$[blk], $\overline{K}^n_{\rm bin}$[blk], $\underline{K}^n_{\rm bin}$[blk] $\leftarrow$ Bin_Conversion()
    hash_verified[blk] $\leftarrow$ Verification()
    if is_hash_verified[blk] **then**
        $\widehat{S}$[blk] $\leftarrow$ Concatenate($\widehat{K}^n_{\rm bin}$[blk], $\underline{K}^n_{\rm bin}$[blk])
        S[blk] $\leftarrow$ Concatenate($\overline{K}^n_{\rm bin}$[blk], $\underline{K}^n_{\rm bin}$[blk])
    end if
end for
$p_{\rm EC}$, FER $\leftarrow$ Frame_Error_Rate_Calculation()
$R, r, \tilde{n}, \varepsilon \leftarrow$ Composable_Rate_Calculation()
if $R > 0$ **then**
    for blk $= 1, 2, \ldots p_{\rm EC}n_{\rm bks}$ **do**
        $\hat{\mathbf{S}} \leftarrow$ Append($\widehat{S}$[blk])
        $\mathbf{S} \leftarrow$ Append(S[blk])
    end for
    $\mathbf{K} \leftarrow$ Privacy_Amplification()
end if
Information_Logging()

IV. SIMULATION RESULTS

We are particularly interested in short-range high-rate implementations of CV-QKD, over distances of around 5 km in standard optical fiber. Even in this regime of relatively high SNR, to get a positive value for the composable secret key rate, we need to consider a block size N of the order of at least 10^5. The choice of the reconciliation efficiency β is also important, as a positive rate cannot be achieved when the value of β is too low.

Sample parameters for a positive R are given in Table IV. Alice's signal variance μ is chosen to achieve a target high value of SNR (e.g., SNR $= 12$ in Figs. 3 and 4). As we see from Fig. 3, positive values for the composable secret key rate are indeed achievable for block sizes with $N > 10^5$ and, as expected, the key rate grows as the block size increases, while all of the simulations attained $p_{\text{EC}} \geq 0.95$. The numerical values of the rate can be considered to be high, since a key rate of 10^{-1} bits/use corresponds to 500 kbits/sec with a relatively slow clock of 5 MHz. Fig. 4 implies that high rates can be achieved even with fewer total states Nn_{bks}, when a large block size, e.g. $N = 250000$, is fixed and the number of blocks n_{blk} varies instead. Note that having an adequately large block size is much more beneficial in obtaining a positive R than having more blocks of smaller sizes. Having fewer total states also achieves faster performance, as seen in Fig. 11.

In Fig. 5, we also show the behavior of the composable secret key rate versus distance L expressed in km of standard optical fiber. We adopt the input parameters specified in Table IV and we use blocks of size $N = 2 \times 10^5$, with reconciliation efficiency β taking values from 90.25% to 92.17%. As we can see from the figure, high rates (around 0.5 bits/use) can be achieved at short distances ($L = 1$ km), while a distance of $L = 7$ km can yield a rate of about 0.004 bits/use.

In Fig. 6, we analyze the robustness of the protocol with respect to the amount of untrusted excess noise in the quantum communication channel (even though this parameter may also include any other imperfection coming from the experimental setup). As we can see from the figure, positive key rates are achievable for relatively high values of the excess noise ($\xi = 0.08$).

Figs. 7, 8 and 9 explore different quantities of interest (FER, rate, and EC rounds respectively) as a function of the SNR and for various choices of the number p of discretization bits. The parameters used in the simulations are given in Table IV, where μ is variable and adapted to attain the desired SNR. In particular, in Figs. 8 and 9, the reconciliation efficiency β (shown in Table V) is chosen according to the following rationale: (i) because a regular LDPC code only achieves a specific value of R_{code}, β is chosen so that R_{code} from Eq. (79) matches R_{code} of a regular LDPC code with high numerical accuracy; (ii) β is high enough so that a positive key rate can be achieved for various values of p for the same SNR; and (iii) β is low enough so that a limited number

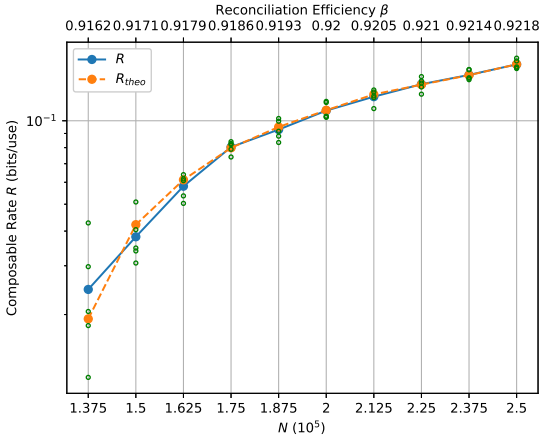

FIG. 3: Composable secret key rate R (bits/use) versus the block size N for SNR $= 12$. We compare the rate of Eq. (89) from five simulations (green points) and their average (blue line) with the theoretical rate of Eq. (92) (orange line), where the theoretical guesses for $\tilde{\beta}$ and $\tilde{p}_{\text{EC}}$ are chosen compatibly with the simulations. For every simulation, $\tilde{p}_{\text{EC}} = p_{\text{EC}}$ has been set. All simulations have achieved $p_{\text{EC}} \geq 0.95$. The step of N is 12500. The values of the reconciliation efficiency β are shown on the top axis and are chosen so as to produce $R_{\text{code}} \approx 0.875$. See Table IV for the list of input parameters used in the simulations.

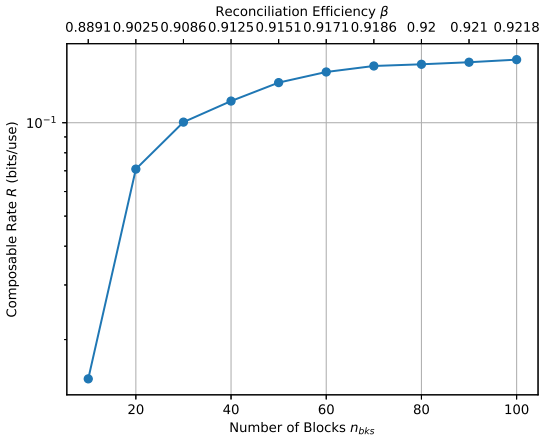

FIG. 4: Composable secret key rate R (bits/use) versus the number of blocks n_{bks} for SNR $= 12$. The step of n_{bks} is 10. The individual block size is fixed and equal to $N = 2.5 \times 10^5$. Every point represents the average value of R, which is obtained after 5 simulations. All simulations have achieved $p_{\text{EC}} \geq 0.95$. The values of the reconciliation efficiency β are shown on the top axis and are chosen so as to produce $R_{\text{code}} \approx 0.875$. See Table IV for the list of input parameters used in the simulations.

of EC rounds exceeds the iteration limit $\text{iter}_{\max}$ (if β is too high, this limit is exceeded and FER increases or can even be equal to 1, meaning that no block is correctly decoded).

Fig. 7 shows the FER for different values of the SNR. As seen, the FER is higher for lower SNRs and quickly declines even with a small increase of the SNR. Note that every simulation, which was executed to produce the

Parameter	Value (Fig. 3)	Value (Fig. 4)	Value (Fig. 5)	Value (Fig. 6)	Value (Fig. 7)	Value (Figs. 8-9)
L	5	5	variable	4	5	5
A	0.2	0.2	0.2	0.2	0.2	0.2
ξ	0.01	0.01	0.01	variable	0.01	0.01
η	0.8	0.8	0.8	0.85	0.8	0.8
v_{el}	0.1	0.1	0.1	0.05	0.1	0.1
n_{bks}	100	variable	100	100	100	100
N	1.375-2.5 $\times 10^5$	2.5×10^5	2×10^5	2.5×10^5	2×10^5	2.5×10^5
M	$0.1 n_{\mathrm{bks}} N$	$0.1 n_{\mathrm{bks}} N$	$0.1 n_{\mathrm{bks}} N$	$0.1 n_{\mathrm{bks}} N$	$0.1 n_{\mathrm{bks}} N$	$0.1 n_{\mathrm{bks}} N$
p	7	7	7	7	7	variable
q	4	4	4	4	4	4
α	7	7	7	7	7	7
$\mathrm{iter}_{\max}$	100	100	150	200	150	150
$\varepsilon_{\mathrm{PE,\ s,\ h,\ \ldots}}$	2^{-32}	2^{-32}	2^{-32}	2^{-32}	2^{-32}	2^{-32}
μ	≈ 21.89	≈ 21.89	20	25	variable	variable

TABLE IV: The input parameters for the simulations.

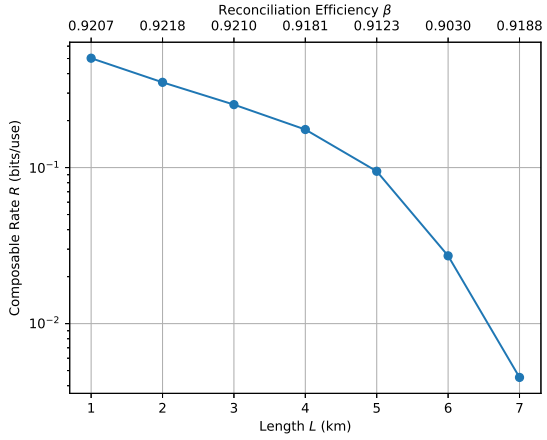

FIG. 5: Composable secret key rate R (bits/use) versus the channel length L (km). Here, we use $N = 2\times10^5$. Every point represents the average value of R, which is obtained after 5 simulations. All simulations have achieved $p_{\mathrm{EC}} \geq 0.95$. The values of the reconciliation efficiency β are shown on the top axis. Other parameters are taken as in Table IV.

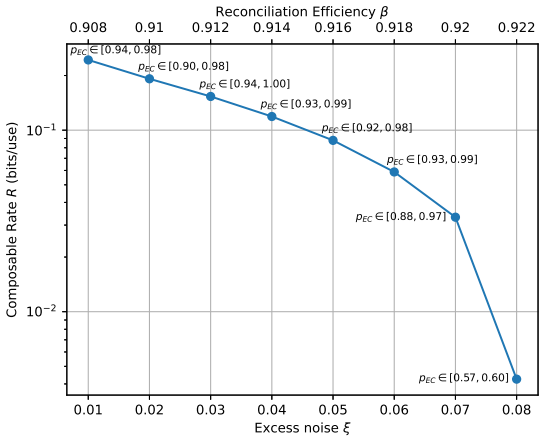

FIG. 6: Composable secret key rate R (bits/use) versus the excess noise ξ. Every point represents the average value of R, which is obtained after 5 simulations. The minimum and maximum values achieved for p_{EC} fall within the interval displayed next to each point. The values of the reconciliation efficiency β are shown on the top axis and are chosen so as to produce $R_{\mathrm{code}} \approx 0.913$. Other parameters are taken as in Table IV.

particular data, returned a positive key rate (the highest FER attained was FER = 0.95 for SNR = 11.725). This result suggests that when N is adequately large, a positive R can be achieved even with a minimal number of correctly decoded and verified blocks. The plot also shows the FER for the same simulations, if the maximum iteration limit had instead been $\mathrm{iter}_{\max} = 100$. In the case of SNR = 11.725, if we had set $\mathrm{iter}_{\max} = 100$, a positive R would not have been realised for some simulations.

Fig. 8 shows the composable key rate R versus SNR for different discretization values p, while keeping the value of q constant and equal to 4 (see the list of parameters in Table IV). As observed, for fixed values of SNR and β, the lower the p is, the higher the rate R is. For every SNR and β, there is a maximum value for p able to achieve a positive R. For example, for SNR = 6 and $\beta \approx 0.8588$ ($R_{\mathrm{code}} \approx 0.75$) a positive R is impossible to achieve with $p \geq 8$. For SNR = 7 and $\beta \approx 0.8775$ ($R_{\mathrm{code}} \approx 0.777$), a positive R is infeasible with $p \geq 9$. The key rate improvement owed to smaller values of p relies on the fact that a smaller amount of bits $d = p - q$ are declared publicly, while the protocol maintains a good EC performance thanks to a sufficiently large number of EC iterations. On the other hand, by increasing p for a fixed q, we increase the number of the public d-bits assisting the LDPC decoding via the sum-product algorithm. This means that the EC step is successfully terminated in fewer rounds.

SNR	$\beta_{p=7}$	$\beta_{p=8}$	$\beta_{p=9}$	R_{code}	d_c
6	0.8588			0.75	8
7	0.8788	0.8775		0.777	9
8	0.8868	0.8865	0.8864	0.8	10
9_{a}	0.89	0.8897	0.8896	0.818	11
9_{b}	0.9265	0.9262	0.9261	0.833	12
10	0.9194	0.9190	0.9189	0.846	13
11_{a}	0.9116	0.9113	0.9113	0.857	14
11_{b}		0.9327	0.9326	0.866	15
12	0.9218	0.9215	0.9214	0.875	16

TABLE V: The chosen reconciliation efficiency β for each SNR of Figs. 8 and 9, together with its respective code rate R_{code} and the row weight d_c of the LDPC code. A missing value for the reconciliation efficiency implies that the returned composable key rate will most likely be negative under the specified values. The column weight d_v remains constant and equal to 2 for all simulations.

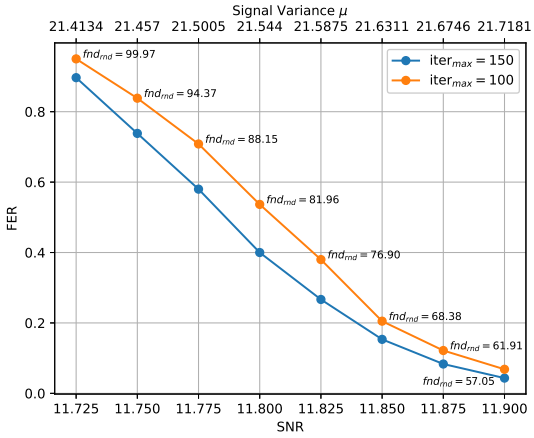

FIG. 7: FER versus SNR for $p = 7$. The FER is compared for the same simulations, when the maximum number of EC iterations is $\text{iter}_{\max} = 150$ (blue line) and when $\text{iter}_{\max} = 100$ (orange line). Every point represents the average value of FER, which is obtained after 6 simulations. The step of the SNR is 0.025. The values of the reconciliation efficiency β are chosen so as to produce $R_{\text{code}} \approx 0.875$. The signal variance μ that was used to achieve the respective SNR is displayed on the top axis with an accuracy of 4 decimal digits. The average number of iterations fnd_{rnd} needed to decode and verify a block is displayed for every point next to their respective points. The other parameters are constant and listed in Table IV. We observe that a slight increase of μ causes the FER to decline rapidly.

In Fig. 9, we plot the average number of EC rounds fnd_{rnd} required to decode a block versus the SNR, for different values of p. For a larger value of p, fewer decoding rounds are needed. This does not only make the decoding faster, but, depending on the specified $\text{iter}_{\max}$, it also gives the algorithm the ability to achieve a lower FER. Thus, a higher p can potentially achieve a better p_{EC}, while a smaller p may return a better R (assuming that $\text{iter}_{\max}$ is large enough). Therefore, at any fixed SNR and $\text{iter}_{\max}$, one could suitably optimize the protocol over the number of discretization bits p.

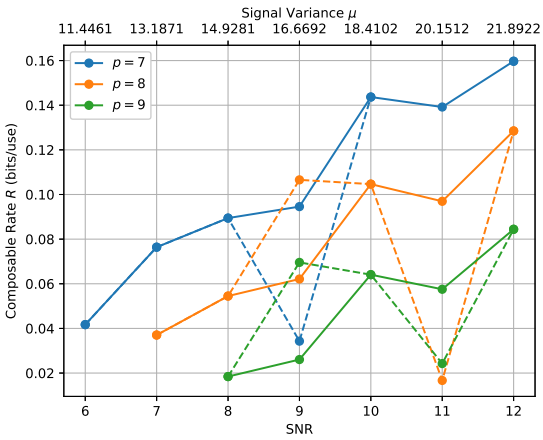

FIG. 8: Composable secret key rate R versus SNR for discretization bits $p = 7, 8, 9$. The chosen reconciliation efficiency β, for each value of the SNR, is shown in Table V. For SNRs $= 9$ and 11, the solid lines follow the values of the entries 'a' of Table V, while the dashed lines describe the 'b' cases. We observe that, for lower values of p (at a fixed $q = 4$), we obtain higher rates for the corresponding SNR. The signal variance μ that was used to achieve the respective SNR is displayed on the top axis with an accuracy of 4 decimal digits. Other parameters are chosen as in Table IV.

V. CONCLUSIONS

In this manuscript we have provided a complete procedure for the post-processing of data generated by a numerical simulation (or an equivalent experimental implementation) of a Gaussian-modulated coherent-state CV-QKD protocol in the high SNR regime. The procedure goes into the details of the various steps of parameter estimation, error correction and privacy amplification, suitably adapted to match the setting of composable finite-size security. Together with the development of the theoretical tools and the corresponding technical details, we provide a corresponding Python library that can be used for CV-QKD simulation/optimization and for the realistic post-processing of data from Gaussian-modulated CV-QKD protocols.

Acknowledgements

A. M. was supported by the Engineering and Physical Science Research Council (EPSRC) via a Doctoral Training Partnership EP/R513386/1.

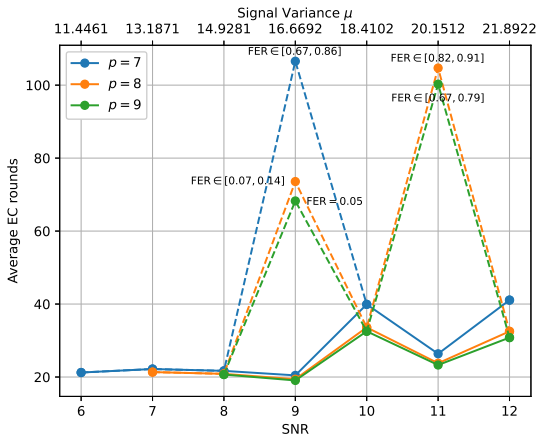

FIG. 9: Average EC rounds $\mathrm{fnd}_{\mathrm{rnd}}$ needed to decode a frame versus the SNR for $p = 7$, $p = 8$ and $p = 9$. A round is registered only if the frame passes the verification step. The chosen reconciliation efficiency β for each value of the SNR is shown in Table V. For SNR $= 9$ and SNR $= 11$ specifically, the solid lines respectively follow the values of the entry 9_{a} and 11_{a} of Table V, while the dashed lines describe the 9_{b} and 11_{b} cases. For the 'b' cases, the FER is reported next to the respective values. The signal variance μ that was used to achieve the respective SNR is displayed on the top axis with an accuracy of 4 decimal digits. Other parameters are chosen as in Table IV.

Appendix A: Alternative formulas for PE

One may define the estimator for the square-root transmissivity $\tau = \sqrt{\eta T}$ as follows

$$\widehat{\tau} = \frac{\sum_{i=1}^{m} x_i y_i}{\sum_{i=1}^{m} x_i^2} \simeq \frac{1}{m\sigma_x^2} \sum_{i=1}^{m} x_i y_i. \tag{A1}$$

We then calculate its variance

$$\begin{aligned} \mathrm{Var}(\widehat{\tau}) =& \frac{\mathrm{Var}\left(\sum_{i=1}^{m} x_i y_i\right)}{m^2(\sigma_x^2)^2} = \frac{\mathrm{Var}(xy)}{m(\sigma_x^2)^2} \\ =& \frac{2}{m}\tau^2 + \frac{\sigma_z^2}{m\sigma_x^2} := \sigma_\tau^2. \end{aligned} \tag{A2}$$

Thus the worst-case estimator for the transmissivity $T = \tau^2$ will be given by

$$\begin{aligned} T_m =& \frac{(\tau - w\sigma_\tau)^2}{\eta} = \frac{\left(\sqrt{\eta T} - w\sqrt{\frac{2}{m}\eta T + \frac{\sigma_z^2}{m\sigma_x^2}}\right)^2}{\eta} \\ =& \frac{\eta T - 2w\sqrt{\eta T}\sqrt{\frac{2}{m}\eta T + \frac{\sigma_z^2}{m\sigma_x^2}}}{\eta} + \mathcal{O}(1/m) \\ \simeq& T\left(1 - 2w\sqrt{1/m}\sqrt{2 + \frac{\sigma_z^2}{\eta T \sigma_x^2}}\right). \end{aligned} \tag{A3}$$

The expression above is the same as the one derived in the main text via Eq. (51)

One may derive a less stringent estimator by assuming the approximation $\sum_{i=1}^{m} x_i^2 \simeq m\sigma_x^2$, meaning that a sample of size m from the data is close enough to reproduce the theoretical variance σ_x^2. In such a case, one may write

$$\begin{aligned} \widehat{\tau} \simeq& \frac{1}{m\sigma_x^2} \sum_{i=1}^{m} x_i(\tau x_i + z_i) = \frac{1}{m\sigma_x^2}\left(\tau \sum_{i=1}^{m} x_i^2 + \sum_{i=1}^{m} x_i z_i\right) \\ \simeq& \frac{1}{m\sigma_x^2}\left(\tau m\sigma_x^2 + \sum_{i=1}^{m} x_i z_i\right) = \tau + \frac{\sum_{i=1}^{m} x_i z_i}{m\sigma_x^2} \end{aligned} \tag{A4}$$

Therefore the variance is now given by

$$\mathrm{Var}(\widehat{\tau}) = \frac{\mathrm{Var}\left(\sum_{i=1}^{m} x_i z_i\right)}{m^2(\sigma_x^2)^2} \tag{A5}$$

$$= \frac{\mathrm{Var}\,(xz)}{m(\sigma_x^2)^2} = \frac{\sigma_z^2}{m\sigma_x^2} := (\sigma_\tau')^2, \tag{A6}$$

yielding the worst-case parameter

$$\begin{aligned} T_m' =& \frac{(\tau - w\sigma_\tau')^2}{\eta} \\ \simeq& T\left(1 - 2w\sqrt{1/m}\sqrt{\sigma_z^2/(\eta T \sigma_x^2)}\right). \end{aligned} \tag{A7}$$

We then observe that the relation in Eq. (A3) gives a more pessimistic value for the worst-case transmissivity due to an extra term equal to 2 appearing in the square root term $\sqrt{2 + \sigma_z^2/(\eta T \sigma_x^2)}$ which is missing in Eq. (A7). In our main text we assume the most conservative choice corresponding to the estimator in Eq. (A3).

Appendix B: Calculations in $\mathcal{GF}(q)$

A Galois field is a field with finite number of elements. A common way to derive it is to take the modulo of the division of the integers over a prime number p. The order of such a field $q = p^k$ (with k being a positive integer) is the number of its elements. All the Galois fields with the same number of elements are isomorphic and can be identified by $\mathcal{GF}(q)$. A special case is the order $q = 2^k$. In a field with such an order, each element is associated with a binary polynomial of degree no more than $k - 1$, i.e. the elements can be described as k-bit strings where each bit of the string corresponds to the coefficient of the polynomial at the same position. For instance, for the element 5 of $\mathcal{GF}(2^3)$ we have

$$101 \to x^2 + 1.$$

This is instructive on how the operations of addition and multiplication are computed in such a field. For example, the addition of 5 and 6 is made in the following way

$$101 + 110 \to (x^2 + 1) + (x^2 + x) = \underbrace{\overbrace{(1+1)}^{0} x^2 + x + 1}_{011 \to 3}.$$

As the field is finite, one can also perform the addition using a precomputed matrix. For instance, for $\mathcal{GF}(2^3)$, we have

$$\mathbf{A}_3 = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 0 & 3 & 2 & 5 & 4 & 7 & 6 \\ 2 & 3 & 0 & 1 & 6 & 7 & 4 & 5 \\ 3 & 2 & 1 & 0 & 7 & 6 & 5 & 4 \\ 4 & 5 & 6 & 7 & 0 & 1 & 2 & 3 \\ 5 & 4 & 7 & 6 & 1 & 0 & 3 & 2 \\ 6 & 7 & 4 & 5 & 2 & 3 & 0 & 1 \\ 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{pmatrix}. \tag{B1}$$

Subtraction between two elements of $\mathcal{GF}(2^k)$ gives the same result as addition, making the two operations equivalent. Multiplication is more complicated, especially when the result is a polynomial with a degree larger then $k-1$. For example, in $\mathcal{GF}(2^3)$, 7×6 is calculated as

$$\begin{aligned} 111 \times 110 \to (x^2+x+1) \times (x^2+x) \\ = x^4+x^3+x^3+x^2+x^2+x = x^4+x. \end{aligned} \tag{B2}$$

Because we have a degree 4 polynomial, we need to take this result modulo an irreducible polynomial of degree 3, e.g., x^3-x+1. Thus, we have

$$(x^4+x \bmod x^3-x+1) = x^2 \to 100 \to 4, \tag{B3}$$

where the operation can be made by adopting a long division with exclusive OR [37]. Instead, as seen in addition, multiplication can be performed by using a precomputed matrix. For instance, in $\mathcal{GF}(2^3)$, the results are specified by the following matrix

$$\mathbf{M}_3 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 2 & 4 & 6 & 3 & 1 & 7 & 5 \\ 0 & 3 & 6 & 5 & 7 & 4 & 1 & 2 \\ 0 & 4 & 3 & 7 & 6 & 2 & 5 & 1 \\ 0 & 5 & 1 & 4 & 2 & 7 & 3 & 6 \\ 0 & 6 & 7 & 1 & 5 & 3 & 2 & 4 \\ 0 & 7 & 5 & 2 & 1 & 6 & 4 & 3 \end{pmatrix}. \tag{B4}$$

Appendix C: LDPC decoding

1. Updating the likelihood function

Let us assume a device where its output is described by the random variable X taking values x according to a family of probability distributions $\mathcal{P}(X;\theta)$ parametrized by θ. Given the sampled data string X_i for $i=1,\dots,n$ from this distribution, one can build a string of data X^n and define the likelihood of a specific parameter θ describing the associated probability distribution as

$$\mathcal{L}(\theta|X^n) = P(X^n|\theta) = \prod_{i=1}^{n} P(X_i|\theta), \tag{C1}$$

where $P(X^n|\theta)$ is the conditional probability for a specific X^n to come out of the device given that its distribution is described by θ and the its outcome of the device is i.i.d. following $\mathcal{P}(X;\theta)$. Intuitively, a good guess $\widehat{\theta}$ of the parameter θ would be the argument θ^* of the maximization of the likelihood function over θ. Using Bayes' rule, we may write

$$P(X^n|\theta) = \frac{P(X^n)}{P(\theta)} P(\theta|X^n), \tag{C2}$$

and observe that $P(X^n)$ is not dependent on θ and $P(\theta)$ is considered uniform (thus independent of θ). Therefore, one may maximize $P(\theta|X^n)$ instead. Furthermore, for the simplification of the later discussion one may express the previous probability as a function being only dependent from the parameter θ (considering the data X_i as constants), namely

$$F(\theta) = P(\theta|X^n). \tag{C3}$$

Let us now consider the case where we have a vector of parameters (variables) $\vec{\theta} = (\theta_1,\dots,\theta_n)$ describing the distribution $\mathcal{P}(X|\boldsymbol{\theta})$. Respectively, one can define the probability

$$F(\vec{\theta}) = F(\theta_1,\dots,\theta_n), \tag{C4}$$

and its marginals

$$F(\theta_i) = \sum_{k\neq i} F(\theta_1,\dots,\theta_k,\dots,\theta_n). \tag{C5}$$

Let us now assume that there are certain constraints that $\vec{\theta}$ should satisfy which are summarized by a system of m linear equations (checks) $\mathbf{H}\vec{\theta} = \vec{z}$, where $\mathbf{H}$ is an $m \times n$ matrix. In particular, there are m equations that the θ_i should satisfy in the form of

$$\sum_i \mathbf{H}_{ji}\theta_i = z_j \quad \text{for} \quad j = 1,\dots,m. \tag{C6}$$

For instance, when $\vec{z} = (3,1,2)$, the matrix in Table VI gives the following three equations [38]:

$$3\theta_3 + \theta_5 = 3, \tag{C7}$$
$$2\theta_1 + \theta_4 = 1, \tag{C8}$$
$$\theta_2 + 2\theta_4 + 3\theta_5 = 2. \tag{C9}$$

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 3 & 0 & 1 \\ 2 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 2 & 3 \end{bmatrix}$$

TABLE VI: An example for a $l \times n$ parity check matrix with values in $\mathcal{GF}(2^2)$ for $l = 3$ checks (check nodes) and $n = 5$ transmitted signals (variable nodes). For this matrix, the assumptions of a regular code explained in Sec. II D are not valid and it is used only as a toy model for the convenience of the description for the sum-product algorithm.

Then one needs to pass from the (a priori) probability distribution of Eq. (C4) to

$$\widetilde{F}(\vec{\theta}) = F(\vec{\theta}|\mathbf{H}\vec{\theta} = \vec{z}), \tag{C10}$$

and calculate the respective marginals

$$\widetilde{F}(\theta_i) = F(\theta_i|\mathbf{H}\vec{\theta} = \vec{z}). \tag{C11}$$

Algorithm 2 Non-Binary Sum-Product Algorithm

Input: $P(\overline{K}_i|X_i\underline{K}_i), K^l_{\rm sd}$, **Output:** $\widehat{K}^n, {\rm fnd}, {\rm fnd}_{\rm rnd}$

Step 1: Initialization
$\vec{z} \leftarrow K^l_{\rm sd}$
$j, i \leftarrow j, i : \mathbf{H}_{ji} \neq 0$ (Tanner graph creation)
$f_i^{\overline{\kappa}} \leftarrow P(\overline{K}_i = \overline{\kappa}|X_i, \underline{K}_i)$
$q_{ji\overline{\kappa}} \leftarrow f_i^{\overline{\kappa}}$
for iter $= 1, 2, \ldots {\rm iter}_{\max}$ **do**
  Step 2: Horizontal Step
  $r_{ji\overline{\kappa}} \leftarrow \sum_{s,t:\ s+t=z_j-\mathbf{H}_{ji}\overline{\kappa}} {\rm Pr}\left[\sigma_{j(i-1)} = s\right] {\rm Pr}\left[\rho_{j(i+1)} = t\right]$
  Step 3: Vertical Step
  $q_{ji}^{\overline{\kappa}} \leftarrow \alpha_{ji} f_i^{\overline{\kappa}} \prod_{m\setminus j} r_{mi\overline{\kappa}}$, $\alpha_{ji} : \sum_{\overline{\kappa}=0}^{2^q-1} q_{ji\overline{\kappa}} = 1$
  Step 4: Tentative Decoding
  $\widehat{K}_i \leftarrow \arg\max_{\overline{\kappa}} f_{i\overline{\kappa}} \prod_j r_{ji\overline{\kappa}}$
  if $\mathbf{H}\widehat{K}^n = \vec{z}$ **then**
    return $\widehat{K}^n, {\rm fnd}_{\rm rnd}, {\rm fnd} \leftarrow$ True
  end if
  if iter $=$ iter$_{\max}$ **then**
    return fnd $\leftarrow$ False
  end if
end for

2. Sum-product algorithm

The sum-product algorithm uses the intuition of the previous analysis to efficiently calculate the marginals

$$\widetilde{F}(\overline{K}_i) = F(\overline{K}_i|\mathbf{H}\overline{K}^n = K^l_{\rm sd}) \tag{C12}$$

of Eq. (C11) for $\theta_i := \overline{K}_i$, $\vec{\theta} := \overline{K}^n$, $\vec{z} := K^l_{\rm sd}$ and the a priori marginal probabilities $F(\overline{K}_i = \overline{\kappa}) = P(\overline{\kappa}|X_i\underline{K}_i)$, calculated in Eq. (66). To do so, it associates a Tanner (factor) graph to the matrix $\mathbf{H}$ and assumes signal exchange between its nodes. More specifically, the graph consists of two kinds of nodes: n variable nodes representing the parameters (variables) $\overline{K}_i$ and m check nodes representing the linear equations (checks) described by $\mathbf{H}\overline{K}^n = K^l_{\rm sd}$. Then, for each variable i that participates in the jth equation, there is an edge connecting the relevant nodes. At this point, we present an example of such a Tanner (factor) graph in Fig. 10(a), based on the matrix $\mathbf{H}$ in Table VI. The signal sent from the variable node i to a factor node j is called $q_{ji\overline{\kappa}}$ and is the probability that the variable $\overline{K}_i = \overline{\kappa}$ and all the linear equations

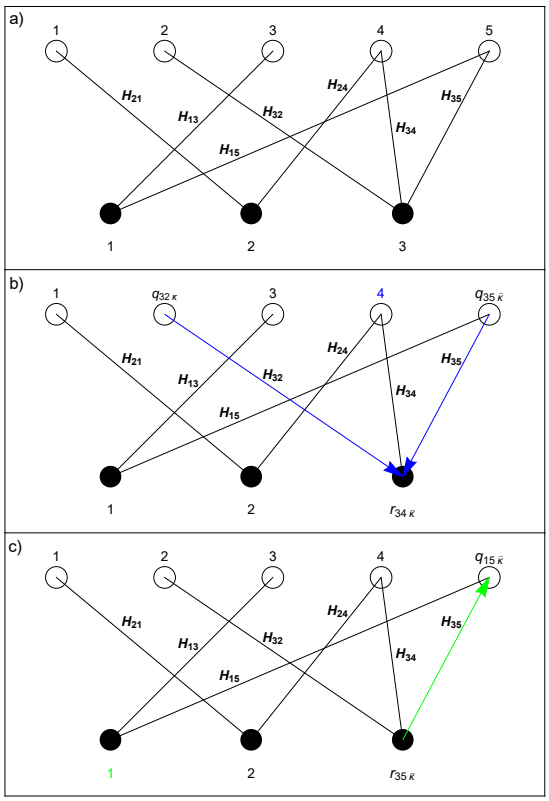

FIG. 10: a) Tanner graph of the parity check matrix of Table VI. The variable (output) nodes (white disks) are connected with the check (syndrome) nodes (black disks) when $\mathbf{H}_{ji} \neq 0$. b) One instance of the horizontal step (Step 2) of Algorithm 2. Here, the signal probability $r_{34\overline{\kappa}}$ is updated for all the $\overline{\kappa} \in \mathcal{GF}(2^2)$ from the contribution (blue arrows) of the rest of the neighbour variable nodes of check node 3, apart from the variable node 4 (node in blue). This update will be repeated in the same step for all the variable nodes, i.e., $r_{32\overline{\kappa}}$ and $r_{35\overline{\kappa}}$ will be calculated too. The same procedure will be followed for syndrome nodes 1 and 2 before the algorithm passes to the horizontal step. This description provides the conceptual steps to derive the desirable result. Practically, the algorithm follows a more complex path, e.g., calculates probabilities of partial sums. However, this path gives an advantage in terms of efficient calculations. c) An instance of the horizontal step (Step 3) of Algorithm 2. Here the probability $q_{15\overline{\kappa}}$ is updated for all the $\overline{\kappa} \in \mathcal{GF}(2^2)$. It is updated only from the contribution of syndrome node 3 (green arrow), while node 1 (node in green) is not participating. This update will happen for all the syndrome nodes, namely $q_{35\overline{\kappa}}$ will be calculated too. It will be repeated also for all variable nodes, before the tentative decoding (Step 4) is going to start.

are true, apart from equation j. The signal sent from the check node j to the variable node i is called $r_{ji\overline{\kappa}}$ and is equal with the probability of equation j to be satisfied, if the variable $\overline{K}_i = \overline{\kappa}$. Note that, based on these definitions, the marginals of Eq. (C12) are given by

$$\widetilde{F}(\overline{K}_i = \overline{\kappa}) = q_{ji\overline{\kappa}} r_{ji\overline{\kappa}}, \tag{C13}$$

for any equation j that the variable i takes part in.

In particular, in each iteration, the algorithm updates the $r_{ji\overline{\kappa}}$ (horizontal step) through the signals of the neighbour variable nodes apart from the signal from node i by the following rule: given a vector $\overline{K}^n$ where its ith element is equal to $\overline{K}_i = \overline{\kappa}$ we have

$$r_{ji\overline{\kappa}} = \sum_{\{i\}} \text{Prob}[K_{\text{sd}\,j}|\overline{K}^n] \prod_{k\in\mathcal{N}(j)\setminus i} q_{jk\overline{K}_k}, \tag{C14}$$

where $\text{Prob}[K_{\text{sd}\,j}|\overline{K}^n]$ takes the value 1, if the check j is satisfied from $\overline{K}^n$, or 0 if it is not. Note that the values of $q_{jk\overline{K}_k}$ are initially updated with the a priori probabilities during the initialization step (see line 5 of Algorithm 2) and that $\mathcal{N}(j)$ are the set of neighbours of the jth check node. An example of such an update is depicted in Fig. 10(b).

In fact, the algorithm takes advantage of the fact that

$$r_{ji\overline{\kappa}} = \text{Prob}\left[\sigma_{j(i-1)} + \rho_{j(i+1)} = K_{\text{sd}\,j} - \mathbf{H}_{ji}\overline{K}_i\right], \tag{C15}$$

where

$$\sigma_{jk} = \sum_{i:i\leq k} \mathbf{H}_{ji}\overline{K}_i, \quad \rho_{jk} = \sum_{i:i\geq k} \mathbf{H}_{ji}\overline{K}_i \tag{C16}$$

are partial sums with different direction running over the jth check. More specifically, Eq. (C15) can be further simplified into a sum of a product of probabilities of the previous partial sums taking specific values by satisfying the jth check, as in line 10 of the pseudocode of Algorithm 2. Then, the algorithm updates the $q_{ji\overline{\kappa}}$ through the signals coming from the neighbour check nodes apart from node j, as depicted for the example in Fig. 10(c). The rule to do so is given in line 12 of the pseudocode (vertical step). Finally, in the tentative decoding step, the algorithm takes the product of $q_{ji\overline{\kappa}}$ and $r_{ji\overline{\kappa}}$, then calculates and maximizes the marginal of Eq. (C13) over $\overline{\kappa}$. The arguments $\widehat{K}_i$ of this maximization of every marginal create a good guess $\widehat{K}^n$ for $\overline{K}^n$. In the next iteration, the algorithm follows the same steps, using the previous $q_{ij\overline{\kappa}}$ to make all the updates.

Appendix D: Toeplitz matrix calculation with Fast-Fourier Transform

The time complexity of the dot product between a Toeplitz matrix $\mathbf{T}$ and a sequence $\mathbf{x}$ is $O(\tilde{n}^2)$. This complexity can be reduced to $O(\tilde{n}\log\tilde{n})$ using the definition of a circulant matrix and the FFT. A circulant matrix $\mathbf{C}$ is a special case of the Toeplitz matrix, where every row of the matrix is a right cyclic shift of the row above it [39]. Such a matrix is always square and is completely defined by its first row $\mathbf{C}_{\text{def}}$.

The steps are as follows [40]:

- The Toeplitz matrix is reformed into a circulant matrix by merging its first row and column together. Since the former has dimensions $\tilde{n}\times r$, where $\tilde{n}$ is the privacy amplification block length and r is the length of the final key, the length of the definition of the latter becomes $\tilde{n}+r-1$.
- The decoded sequence to be compressed is extended, as $r-1$ zeros are padded to its end. The length of the new sequence $\mathbf{S}_{\text{ext}}$ is now equal to the length of the circulant matrix definition.
- To efficiently calculate the key, an optimized multiplication is carried out as

$$\mathfrak{F}^{-1}[\mathfrak{F}(\mathbf{S}_{\text{ext}}) * \mathfrak{F}(\mathbf{C}_{\text{def}})] \tag{D1}$$

  where $\mathfrak{F}$ represents the FFT and $\mathfrak{F}^{-1}$ stands for the inverse FFT. Because of the convolution theorem, the $*$ operator signifies the Hadamard product and therefore element-wise multiplication can be performed.
- As the key format is required to be in bits, the result of the inverse FFT is taken modulo 2.
- The key $\mathbf{K}$ is constituted by the first r bits of the resulting bit string of length $\tilde{n}+r-1$.

Appendix E: Software benchmarks

The benchmarks for the entire runtime duration and the peak memory consumption, with regards to different sizes for the N and n_{bks} variables, are portrayed in Fig. 11. Around 95% of the runtime is ascribed to the duration of the non-binary sum-product algorithm, while the heavy memory load is predominantly because of the privacy amplification stage. The slow speed of the decoding stage is justified, as the non-binary sum-product algorithm is complex in nature. In addition, in order to achieve a positive composable key rate, the block sizes have to be sufficiently large ($N > 10^5$) and an adequate number of blocks has to be present as well.

For such a computational task, we employed the Interactive Research Linux Service of University of York, whose specifications are noted in Table VII. Nevertheless, the software is able to run on a conventional computer as well; however, the speed will be significantly diminished. To provide algorithmic speedups we used various techniques, which include, but are not limited to the Numba library, parallelization, the use of dictionary structures (whose lookup time complexity is $\mathcal{O}(1)$) and precomputed tables for the Galois field computations.

An advantage of the sum-product algorithm is that it is parallelizable. Therefore, possessing more processing cores is beneficial in terms of speed and, consequently, such projects are often carried out on Graphical Processing Units (GPUs) [21] because of their superior number of cores compared to Central Processing Units (CPUs). To provide massive compatibility, the software is written to target solely CPUs. Future versions of the software may process the error correction stage on a GPU level.

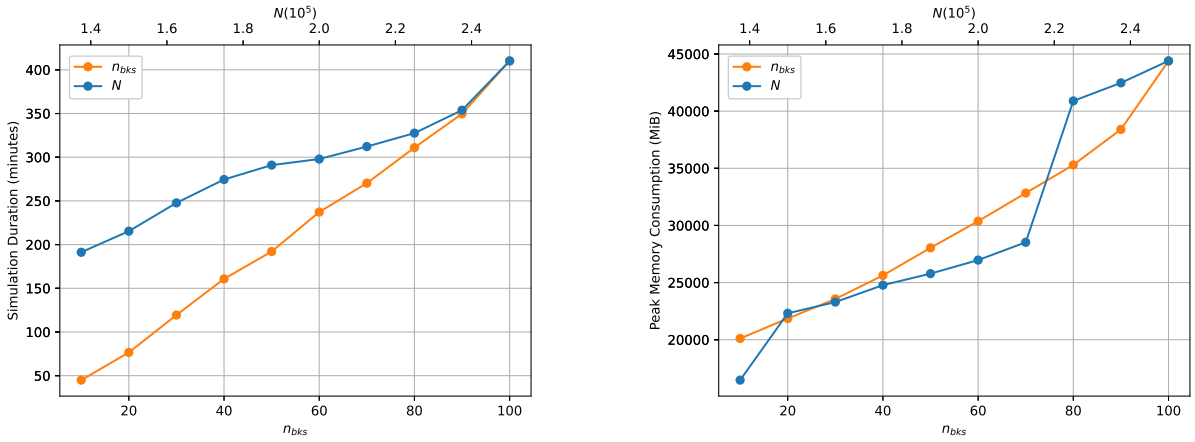

FIG. 11: Computational benchmarks: Entire duration of a simulation (left) and peak memory consumption (right). These are provided with respect to a variable block size N (in blue) and a variable number of blocks n_{bks} (in orange). When N is variable, the number of blocks is constant and equal to $n_{\text{bks}} = 100$. When n_{bks} is variable, the block size is constant and equal to $N = 250000$. The results depicted are based on the simulations of Fig. 3 and Fig. 4, respectively (whose corresponding sets of parameters can be found in Table IV). In order to successfully decode a block, around 40 iteration rounds are required on average.

CPU Model	Intel Xeon E5-2680 v4
CPU Clock Speed	2.60 GHz
Number of Cores	56
RAM	512GB
OS	Ubuntu 20.04
Python Version	3.8

TABLE VII: The specifications of the system, on which the simulations were executed.

In addition, the non-binary sum-product method used in this paper is anachronistic in terms of speed. There exists a newer method, which replaces the stages that demand the most complexity with FFT computations [41]. Future improvements on the algorithm could potentially include this method as well.

Generating a shared secret key $\mathbf{K}$ for a particular set of noise parameters in a quick manner is a matter of optimization of the block size, the number of blocks and the discretization bits. Let us examine the well-studied case of SNR $= 12$. In Sec. IV, it is explained that selecting a small number of blocks with a large block size over a large number of blocks with a smaller block size is beneficial for the key rate. Furthermore, in Fig. 9, it is shown that $p = 8$ provides a reasonable boost in the error-correction speed. On the contrary, choosing $p = 9$ offers little advantage when the average number of iterations is already small; however the key rate sacrifice is large. Under $R_{\text{code}} = 0.875$ and $p = 8$, the algorithm needs around 32.5 iterations on average to decode a block. Taking all the above into consideration, a simulation with the parameters $N = 352000$, $n_{\text{bks}} = 5$ and $p = 8$ can produce a key with a composable key rate of around 10^{-4} bits/use and a delay time of less than 30 minutes.

[1] S. Pirandola, U. L. Andersen, L. Banchi, M. Berta, D. Bunandar, R. Colbeck, D. Englund, T. Gehring, C. Lupo, C. Ottaviani, J. L Pereira, M. Razavi, J. S. Shaari, M. Tomamichel, V. C. Usenko, G. Vallone, P. Villoresi, and P. Wallden, "Advances in quantum cryptography," Adv. Opt. Photon. **12**, 1012–1236 (2020).

[2] J. Park, "The concept of transition in quantum mechanics," Foundations of Physics **1** 23–33 (1970).

[3] W. Wootters, and W. Zurek, "A Single quantum cannot be cloned," Nature **299**, 802–803 (1982).

[4] C. H. Bennett and G. Brassard, "Quantum cryptography: public key distribution and coin tossing," in Proceedings of the International Conference on Computers, Systems & Signal Processing, Bangalore, India, December 1984, pp. 175–179.

[5] D. Bunandar, L. C. G. Govia, H. Krovi, and D. Englund, "Numerical finite-key analysis of quantum key distribution," npj Quantum Inf **6**, 104 (2020).

[6] F. Grosshans and P. Grangier, "Continuous variable quantum cryptography using coherent states," Phys. Rev. Lett. **88**, 057902 (2002).

[7] C. Weedbrook, A. M. Lance, W. P. Bowen, T. Symul, T. C. Ralph, P. K. Lam, "Quantum cryptography without switching," Phys. Rev. Lett. **93**, 170504 (2004).

[8] C. Weedbrook, S. Pirandola, R. García-Patrón, N. J. Cerf, T. C. Ralph, J. H. Shapiro, and S. Lloyd, "Gaussian quantum information," Rev. Mod. Phys. **84**, 621 (2012).

[9] S. Pirandola, R. Laurenza, C. Ottaviani, and L. Banchi, "Fundamental Limits of Repeaterless Quantum Communications," Nat. Commun. **8**, 15043 (2017).

[10] S. Pirandola, C. Ottaviani, G. Spedalieri, C. Weedbrook, S. L. Braunstein, S. Lloyd, T. Gehring, C. S. Jacobsen, and U. L Andersen, "High-rate measurement-device-independent quantum cryptography," Nat. Photon. **9**, 397-402 (2015).

[11] Y. Zhang, Z. Chen, S. Pirandola, X. Wang, C. Zhou, B. Chu, Y. Zhao, B. Xu, S. Yu, and H. Guo, "Long-Distance Continuous-Variable Quantum Key Distribution over 202.81 km of Fiber," Phys. Rev. Lett. **125**, 010502 (2020).

[12] C. Zhou, X. Wang, Y. Zhang, Z. Zhang, S. Yu, and H. Guo, "Continuous-Variable Quantum Key Distribution with Rateless Reconciliation Protocol," Phys. Rev. Applied **12**, 054013 (2019).

[13] The Python library is available on github at the address softquanta/homCVQKD

[14] S. Pirandola, "Limits and Security of Free-Space Quantum Communications," Phys. Rev. Res. **3**, 013279 (2021).

[15] D. J. C. Mackay, "Good error-correcting codes based on very sparse matrices," IEEE Trans. Inform. Theory **45**, 399–431 (1999).

[16] M. C. Davey and D. MacKay, "Low-density parity check codes over GF(q)," IEEE Communications Letters **2**, 165–167 (1998).

[17] D. Huang, D. Lin, C. Wang, W. Liu, S. Fang, J. Peng, P. Huang, and G. Zeng, "Continuous-variable quantum key distribution with 1 mbps secure key rate," Opt. Express **23**, 17511–17519 (2015).

[18] D. K. Lin, D. Huang, P. Huang, J. Y. Peng, and G. H. Zeng, "High performance reconciliation for continuous-variable quantum key distribution with LDPC code," Int. J. Quantum Inform. **13**, 1550010 (2015).

[19] C. Pacher, J. Martinez-Mateo, J. Duhme, T. Gehring, and F. Furrer, "Information Reconciliation for Continuous-Variable Quantum Key Distribution using Non-Binary Low-Density Parity-Check Codes," arXiv:1602.09140.

[20] F. R. Kschischang, B. J. Frey, and H. Andrea Loeliger, "factor graphs and the sum-product algorithm," IEEE Trans. Inform. Theory **47**, 498–519 (2001).

[21] M. Milisevic, "Low-Density Parity-Check Decoder Architectures for Integrated Circuits and Quantum Cryptography," (PhD thesis, University of Toronto, 2017).

[22] T. Tsurumaru and M. Hayashi, "Dual universality of hash functions and its applications to quantum cryptography," IEEE Trans. Info. Theory **59**, 4700–4717 (2013).

[23] Q. Li, B.-Z. Yan, H.-K. Mao, and X.-F. Xue, "High-speed implementation of fft-based privacy amplification on fpga in quantum key distribution," arXiv:1809.07592.

[24] S. Pirandola, S. L. Braunstein, and S. Lloyd, "Characterization of Collective Gaussian Attacks and Security of Coherent-State Quantum Cryptography," Phys. Rev. Lett. **101**, 200504 (2008).

[25] T. M Cover and J. A. Thomas, "Elements of information theory," (Wiley, 2012).

[26] D. S. Slepian and J. K. Wolf, "Noiseless Coding of Correlated Information Sources", IEEE Trans. Inf. Theory **19**, 471–480 (1973).

[27] A. D. Wyner, "Recent results in Shannon theory," IEEE Trans. Inform. Theory **20**, 2-10 (1974).

[28] L. Ruppert, V. C. Usenko, and R. Filip, "Long-distance continuous-variable quantum key distribution with efficient channel estimation", Phys. Rev. A **90**, 062310 (2014).

[29] P. Papanastasiou, C. Weedbrook, and S. Pirandola, "Continuous-variable quantum key distribution in fast fading channels," Phys. Rev. A **97**, 032311 (2018).

[30] D. J. C. MacKay, and R. M. Neal, "Near Shannon limit performance of low density parity check codes," Elect. Lett. **33**, 457-458 (1997).

[31] M. Thorup, "High Speed Hashing for Integers and String," arXiv:1504.06804v9.

[32] A. Antos and I. Kontoyiannis, "Convergence Properties of Functional Estimates for Discrete Distributions," Random Structures & Algorithms **19**, 163 (2001).

[33] Note that α and q also affect the a priori probabilities of the decoding step in Eqs. (66) and (67) which, in turn, affect the p_{EC} for a given number of maximum iterations $\text{iter}_{\max}$.

[34] S. Pirandola, "Composable security for continuous variable quantum key distribution: Trust levels and practical key rates in wired and wireless networks," Phys. Rev. Res. 3, 043014 (2021).

[35] We remark that the formula for the key rate in Eq. (89) is valid under the assumption of collective Gaussian attacks, which is the state-of-the-art highest level of security for the Gaussian-modulated homodyne protocol. Security of the heterodyne protocol [7] could be extended to fully coherent attacks due to the extra symmetry of the protocol under the unitary group provided by the hetorodyne measurement [1]. In order to enforce this symmetry, the gathered data associated with block size N needs to be multiplied by a random orthogonal $N \times N$ matrix. To the best of our knowledge, the complexity for creating such a matrix is $\mathcal{O}(N^2 \log N)$ [36]. While this is certainly possible from a theoretical point of view, it is very challenging to be implemented in practical data processing since the value of the block size is $N > 10^5$. The time overhead involved in this step is extremely long (with current processing power).

[36] G. W. Stewart, "The efficient generation of random orthogonal matrices with an application to condition estimation," SIAM J. Numer. Anal. **17**, 403–409 (1980).

[37] G. L. Mullen and D. Panario, "Handbook of Finite Fields," (CRC Press, 2013).

[38] Note that these equations are valid in $\mathcal{GF}(4)$

[39] R. M. Gray, "Toeplitz and Circulant Matrices: A Review," Foundations and Trends in Communications and Information Theory **2**, 155-239 (2006).

[40] B. Tang, B. Liu, Y. Zhai, C. Wu, and W. Yu, "High-speed and Large-scale Privacy Amplification Scheme for Quantum Key Distribution", Sci. Rep. **9**, 15733 (2019).

[41] L. Safarnejad, M. Sadeghi, "FFT Based Sum-Product Algorithm for Decoding LDPC Lattices," IEEE Communications Letters **16**, 1504-1507 (2012).